

# Asynchronous Signal Passing for Tile Self-Assembly: Fuel Efficient Computation and Efficient Assembly of Shapes

Jennifer E. Padilla<sup>\*</sup>, Matthew J. Patitz<sup>\*\*</sup>, Raul Pena <sup>\*\*\*</sup>, Robert T. Schweller<sup>†</sup>, Nadrian C. Seeman<sup>‡</sup>,  
Robert Sheline <sup>§</sup>, Scott M. Summers<sup>¶</sup>, and Xingsi Zhong <sup>||</sup>

**Abstract.** In this paper we study the power of a model of tile self-assembly in which individual tiles of the system have the ability to turn on or off glue types based on the bonding of other glues on the given tile. This work is motivated by the desire for a system that can effect recursive self assembly, and is guided by the consideration of a DNA origami implementation of four-sided tiles which have a small set of queued up reactions based on DNA strand replacement. We formulate the Signal passing Tile Assembly Model which is based on the model of Padilla, et al. (2011) [21], but is asynchronous in that any action of turning a glue on or off, or attaching a new tile, or the breaking apart of an assembly, may happen in any order. Within this most general case we study tile self-assembly problems that have been addressed within the abstract Tile Assembly Model, as well as other variants of that model, and show that signal passing tiles allow for substantial improvement across multiple metrics. Our first results focus on the tile type efficient assembly of linear structures, which achieves assembly using provably fewer tile types than what is possible in standard tile assembly models by utilizing a recursive assembly process. We then present a system of signal passing tiles that simulate an arbitrary Turing machine which is *fuel efficient* in that only a constant number of tiles are used up per computation step. To the best of our knowledge, this is the first tile self-assembly model that is able to achieve this. Finally, we consider the assembly of the discrete Sierpinski triangle and show that this pattern can be strictly self-assembled within the signal tile passing model. This result is of particular interest in that it is known that this pattern cannot self-assemble within a number of well studied tile self-assembly models.

## 1 Introduction

The Tile Assembly Model (TAM) [35] created a paradigm for computation to be carried out by a physical assembly process that captured the essence of the Wang tiling model [33]. Turing machine simulation within the TAM demonstrated its capacity for universal computation, and many subsequent assembly programs shifted focus to patterns, shapes and structures as the output of tile computation [14, 29, 30, 32]. Many modifications to the standard TAM have been investigated, including several variants that capture the notion of hierarchical assembly [2, 3, 8]. Recent work has introduced the notion of using signaled glue activation to guide hierarchical assembly, enabling recursive self assembly [19, 21].

<sup>\*</sup> Department of Chemistry, New York University, [jp164@nyu.edu](mailto:jp164@nyu.edu) This author's research was supported by National Science Foundation Grant CCF-1117210.

<sup>\*\*</sup> Department of Computer Science, University of Texas - Pan American, [mpatitz@cs.panam.edu](mailto:mpatitz@cs.panam.edu) This author's research was supported in part by National Science Foundation Grant CCF-1117672.

<sup>\*\*\*</sup> Department of Computer Science, University of Texas - Pan American, [nb-raul@live.com](mailto:nb-raul@live.com) This author's research was supported in part by National Science Foundation Grant CCF-1117672.

<sup>†</sup> Department of Computer Science, University of Texas - Pan American, [schwellerr@cs.panam.edu](mailto:schwellerr@cs.panam.edu) This author's research was supported in part by National Science Foundation Grant CCF-1117672.

<sup>‡</sup> Department of Chemistry, New York University, [ned.seeman@nyu.edu](mailto:ned.seeman@nyu.edu) This author's research was supported in part by National Science Foundation Grant CCF-1117210.

<sup>§</sup> Department of Computer Science, University of Texas - Pan American, [b.sheline@gmail.com](mailto:b.sheline@gmail.com) This author's research was supported in part by National Science Foundation Grant CCF-1117672.

<sup>¶</sup> Department of Computer Science and Software Engineering, University of Wisconsin-Platteville, [summerss@uwplatt.edu](mailto:summerss@uwplatt.edu)

<sup>||</sup> Department of Computer Science, University of Texas - Pan American, [zhongxingsi@gmail.com](mailto:zhongxingsi@gmail.com) This author's research was supported in part by National Science Foundation Grant CCF-1117672.

A model is developed here that in the most general way possible, describes a signaled tile self assembly process that enriches the tile assembly paradigm with improved capabilities and allows tile assembly to more closely emulate biological and natural processes.

Signaled glue activation was introduced for the purpose of allowing supertiles to take on new identities or roles once assembled [21], so that supertile interactions as described in hierarchical models such as the 2-HAM [1, 2] can simulate the interactions of individual tiles. In particular, recursive assembly results when supertiles simulate the original tiles of the tile set [21], a strategy outlined here in section 3.3 in a scheme for efficient line assembly. The line assembly program makes use of a random assignment of the fate of a given supertile. Here, random choice of roles (left or right half of the line) are made by random tile binding events. A similar random choice of fate was utilized in the Robinson tiling construction of [21], and is a feature that makes effective use of the random interactions expected in a test tube of well-mixed DNA tiles that might implement the model.

In plausible experimental implementations using DNA, such as the mechanism depicted in Figure 1, glue deactivation can be expected to be as easy to implement as glue activation. Therefore, we utilize this new capability to design a Turing Machine that is fuel efficient, and to implement the strict assembly of the Sierpinski triangle. While on first consideration, glue deactivation might be thought to be on par with negative glues, it is in fact far more easily implemented, and never requires repulsive forces between tiles, a necessity for negative glues that to our knowledge has yet to be implemented.

Glue deactivation is used here to produce a fuel efficient Turing Machine. Unlike with other tile implementations, this Turing Machine does not need to keep using up tiles to copy the state of unchanged positions on the tape. A halting computation produces an output tape, not a transcript of the computation as in traditional 2D tile simulation of a Turing Machine. This makes it easier to view the Turing Machine plus its tape as a developing entity, that by following its input instructions, much as a cell follows its genetic blueprint, goes through a metamorphosis and emerges from a halting computation as a new entity.

Strict self assembly of the Sierpinski triangle is achieved by releasing tiles that are not part of the target structure. This asynchronous process resembles the growth of something such as coral, where the “living” functional part of the system inhabits the growing frontier of the structure, and lays down an enduring structure before dying and being washed away. The growth is unlimited in the sense that there is no prescribed largest entity that can be formed. This construction, unlike the line assembly, still occurs tile-by-tile as in the TAM, but by releasing unwanted tiles results in the otherwise impossible strict assembly of the self-similar Sierpinski triangle. Again, we note that no repulsive forces are required, as no negative glues are used.

The addition of signaled glue activation and deactivation to tile assembly brings it one step closer to emulating biological processes of self assembly, where communication within a developing and growing living organism are crucial to its survival and success. Communication between tiles and throughout assemblies allows them to take on changing roles in hierarchical assembly processes, resulting in production of organized entities that could continue to function as elements in further tile computation. To show that the new model is possible to implement physically as the TAM has been [29], we describe the rudiments of a physical implementation that both inspires and justifies the model, and conclude that the constructions presented in this paper demonstrate not only a more efficient use of materials (Table 1) in certain cases, but also serve to make the model more relevant in a biological context.

## 2 Physical Basis for the Model

The generalized model presented here has been designed to take into consideration a DNA implementation of all aspects of signaled assembly: binding, signaling, and glue activation or deactivation. We envision a physical implementation where Watson-Crick DNA base pairing provides specific glue interactions as has been done before for the aTAM [29, 35]. The model, simplified as it is with non-rotatable tiles and a diagonal glue function, can nonetheless be implemented by complementary DNA strands on free-floating tiles by noting that East-West and North-South pairs are complementary and distinct from each other. The physical body of a tile might be composed entirely from a DNA origami structure [17, 28] which has more room for signal pathways than smaller DNA structures that have been used to implement the passive tiles of the TAM [36]. Toe-hold mediated DNA strand exchange mechanisms [31, 38–40], as shown in Figure 1, provide a basis for a plausible physical implementation of signaling, glue activation and glue deactivation. Signals across individual tiles could potentially be

Linear Assemblies	Tiles	Signals	Temperature	Glue Deactivation
aTAM	n	-	1	-
STAM (Thm.1)	4	$O(\log n)$	1	No.
STAM (Thm.2)	$O(\log n / \log \log n)$	$O(1)$	1	Yes.

Turing Machine	Space	Fuel	Temperature	Tiles	Signals
aTAM ( [26])	$O(\sum S_i)$	$O(S_i)$	2	$O(Q)$	-
3D aTAM ( [7])	$O(\sum S_i)$	$O(S_i)$	1	$O(Q)$	-
Negative Glues ( [13])	$O(S_i)$	$O(S_i)$	2	$O(Q)$	-
STAM (Thm.3)	$O(S_i)$	$O(1)$	1	$O(Q)$	$O(1)$

Sierpinski Triangle	Strict	Tiles	Signals	Temperature	Scale	Glue Deactivation
aTAM ( [27])	No.	7	-	2	1	-
STAM (Thm.5)	Yes.	19	5	1	2	Yes.
STAM (Thm.4)	No.	5	4	1	1	No.

Table 1: Summary of our Results in the context of previous work in the field.

implemented using DNA hybridization cascades such as HCR [11], transducers [31], seesaw gates [25], other mechanisms [38], or by DNA walkers [18, 20, 34, 38]. Details of a plausible DNA origami tile construction are given in [21].

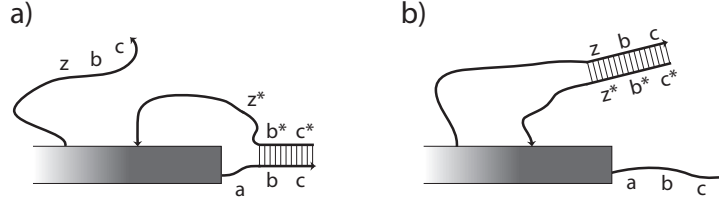


Fig. 1: Activation of a latent glue. Single-stranded DNA is depicted as a line with an arrow at the 3' end and double-stranded DNA is shown as two parallel lines connected by a ladder of base pairs. Letters such as  $a$  and  $y$  indicate short sequences of DNA bases and a star indicates the complementary sequence. Rectangles indicate structures, possibly composed of DNA origami, to which the relevant sticky ends of DNA are attached. a) The strand  $zbc$  is presumed to be unavailable until freed by a previous signaling cascade (not shown). The sequence labeled  $z$  may then bind at the toehold  $z^*$  on the protection strand  $c^*b^*z^*$ , removing it from the glue strand  $abc$  via strand displacement. The protection strand is shown bound only to  $b$  and  $c$ , which is enough to prevent interaction with the complementary tile edge shown as long as no toeholds are free to interact. Alternatively, the entire sequence  $abc$  could be blocked by the protection strand. b) Once the protection strand is removed, the glue strand  $abc$  is active or *on*.

A glue may be considered 'latent' or 'off' if its DNA sticky end is blocked by being bound to a complementary DNA strand. Activation of a latent glue can be done by the toe-hold mediated removal of a protection strand that blocks a sticky end, (Figure 1). Deactivation can be accomplished by a toehold-mediated mechanism that is essentially the reverse of activation, whereby a protection strand can be released in the vicinity of the glue strand, covering up the sticky end used for binding. By adding more toeholds, one could add a finite number of cycles of activation and deactivation, but each would be mediated by a different activation and deactivation strand, and each effectively consumes DNA fuel. Thus, in the model we have limited activation to occurring only once. More instances of a glue can be added in a construction if needed. Please see Section A.1 of the appendix for more details.

### 3 STAM Notation and Model

In this section we define the Signal Tile Assembly Model (STAM), an extension of the 2-Handed Assembly Model (2HAM) [4–6, 9], which itself is an extension of the Tile Assembly Model (TAM) [37]. The STAM is a refined version of the model presented in [22], in which the basic tiles of the TAM are augmented with the ability to receive information, in the form of *signals*, from neighboring tiles in

an assembly, and to pass signals to neighboring tiles. A very important feature of signals is that each signal can only move across any given tile one time - they are not reusable.

The STAM that we define is a highly generalized model which imposes minimal restrictions on aspects such as the speed of signals and orderings of events. This generalized version, while perhaps difficult to create well-behaved constructions within, provides a framework that is intended to be maximally independent of the specific details of potential physical implementations of actual signal tiles, such as those using mechanisms shown in [22]. Valid constructions within this model, such as all of the constructions presented within the following sections of this paper, will therefore also work correctly within more restricted versions of the model.

**Informal Description of the 2HAM** Since the STAM is an extension of the 2HAM, we now give a brief, informal description of the 2HAM.

A *tile type* is a unit square with four sides, each having a *glue* consisting of a *label* (a finite string) and *strength* (a positive integer value). We assume a finite set  $T$  of tile types, but an infinite number of copies of each tile type, each copy referred to as a *tile*. A *supertile* (a.k.a., *assembly*) is a positioning of tiles on the integer lattice  $\mathbb{Z}^2$ . Two adjacent tiles in a supertile *interact* if the glues on their abutting sides are equal (in both label and strength) and *bind* with that shared strength. Each supertile induces a *binding graph*, a grid graph whose vertices are tiles, with an edge between two tiles if they interact and where the weight of that edge is the strength of their bond. The supertile is  $\tau$ -*stable* if every cut of its binding graph has strength at least  $\tau$ . That is, the supertile is stable if at least energy  $\tau$  (i.e. a cut across bonds whose strengths sum to at least  $\tau$ ) is required to separate the supertile into two parts. A *tile assembly system* (TAS) is a pair  $\mathcal{T} = (T, \tau)$ , where  $T$  is a finite tile set (or more generally a finite set of supertiles) and  $\tau$  is the *temperature*, a parameter specifying the minimum binding energy required for a supertile to be stable. Given a TAS  $\mathcal{T} = (T, \tau)$ , a supertile is *producible* if either it is a single tile from  $T$ , or it is the  $\tau$ -stable result of translating two producible assemblies without overlap. A supertile  $\alpha$  is *terminal* if for every producible supertile  $\beta$ ,  $\alpha$  and  $\beta$  cannot be  $\tau$ -stably attached. A TAS is *directed* (a.k.a., *deterministic*, *confluent*) if it has only one terminal, producible supertile. Given a connected shape  $X \subseteq \mathbb{Z}^2$ , a TAS  $\mathcal{T}$  *strictly self-assembles*  $X$  (also *produces*  $X$  *uniquely*) if every producible, terminal supertile places tiles exactly on those positions in  $X$  (appropriately translated if necessary). Given a pattern  $Y \subseteq \mathbb{Z}^2$  (which must not necessarily be connected), a TAS  $\mathcal{T}$  *weakly self-assembles*  $Y$  if every producible, terminal supertile places a subset of tiles  $B \subseteq T$  exactly on those positions in  $Y$  (appropriately translated if necessary). Weak self-assembly can be thought of as using a subset of tile types to “paint a picture” of  $Y$  on a possibly larger canvas of tiles composing a terminal assembly.

**High Level Description of the STAM** In the STAM, tiles are allowed to have sets of glues on each edge (as opposed to only one glue per side as in the TAM and 2HAM). Tiles have an initial state in which each glue is either “**on**” or “**latent**” (i.e. can be switched **on** later). Tiles also each implement a transition function which is executed upon the binding of any glue on any edge of that tile. The transition function specifies a set of glues (along with the sides on which those glues are located) and an action for each: 1. turn that glue **on** (only valid if it is currently **latent**), or 2. turn that glue **off** (valid if it is currently **on** or **latent**). Note that turning a glue **off** breaks any bond that that glue may have formed with a neighboring tile. The transition function defined for each tile type is allowed a unique set of output actions for the binding event of each glue along its edges. As the STAM is an extension of the 2HAM, binding and breaking can occur between pairs of arbitrarily sized supertiles. In order to allow for physical mechanisms which implement the transition functions of tiles but are arbitrarily slower or faster than the average rates of (super)tile attachments and detachments, rather than immediately enacting the outputs of transition functions, each output action is put into a set of “pending actions” which includes all actions which have not yet been enacted for that glue. An STAM system consists of a set of tiles and a temperature value. To define what is producible from such a system, we use a recursive definition of producible assemblies which starts with the initial tiles and then contains any supertiles which can be formed by doing the following to any producible assembly: 1. executing any entry from the pending actions of any one glue within a tile within that supertile (and then that action is removed from the pending set), 2. binding with another supertile if they are able to form a  $\tau$ -stable supertile, or 3. breaking apart into 2 separate supertiles along a cut whose total strength is less than  $\tau$ .

The remainder of this section presents a formal definition of the STAM. In order to help clarify many aspects, a consistent example is provided and referenced throughout.

**Basic Notation** Let  $D$  denote the set of labels  $\{\text{north}, \text{south}, \text{east}, \text{west}\}$ , or  $\{N, S, E, W\}$ .

**Active Glues and Glue States** Let  $\Gamma$  denote a set of *glue types*. A *glue* is an ordered pair  $(g, s)$  where  $g \in \Gamma$  is the glue type and  $s \in \mathbb{Z}$  is the glue *strength*. Note that throughout the remainder of this paper, unless specifically stated, all glues will have strength 1 and as shorthand will be denoted simply by their glue types with the strength omitted.

Let  $Q = \{\text{latent}, \text{on}, \text{off}\}$  be the set of possible *states* for a glue. Intuitively, **on** is the “normal”, active state of a glue, meaning that it is either able to bind or currently bound. A glue in the **latent** state is inactive (and therefore unable to bind), and also has never been **on** (or **off**). A glue in the **off** state is also inactive and unable to bind, but can never be (re)activated. We define an *active glue* as an ordered pair  $(g, q)$  where  $g \in \Gamma$  is a glue type and  $q \in Q$  is a state.

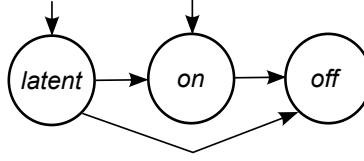


Fig. 2: The valid state transitions for active glues

**Active Labels** As in the original TAM, we define a *label* as an arbitrary string over some fixed alphabet and labels will be used as non-functional (i.e. they don’t participate in tile bindings) means of identifying tile types. Denote as  $\Sigma$  the set of all valid labels. For the self-assembly of patterns (e.g. the weak self-assembly of the Sierpinski triangle), tile labels are the mechanism used for distinguishing between groups of tiles, i.e. those “in” the pattern and those “outside” it. Experimentally, labels can be implemented as DNA loops which protrude above the surfaces of tiles for imaging purposes [16], and thus the motivation exists to allow for the modification of tile labels along with glues. Therefore, we define an *active label* is an ordered pair  $(x, q)$  where  $x \in \Sigma$  and  $q \in Q$ .

**Active Tiles and Transition Functions** A *generalized active tile type*  $t$  is a unit square defined as a 4-tuple  $t = (G, L, \delta, \Pi)$  where  $G : D \rightarrow P(\Gamma \times Q)$  is a function specifying the set of all active glues present on a given side,  $L$  is the set of all active labels,  $\delta : D \times \Gamma \rightarrow P(((D \times \Gamma \times \{\text{on}, \text{off}\}) \cup (\Sigma \times \{\text{on}, \text{off}\})))$  is the *transition function* and  $\Pi$  is a multiset over  $(D \times \Gamma \times \{\text{on}, \text{off}\}) \cup (\Sigma \times \{\text{on}, \text{off}\})$ . A generalized active tile type  $t = (G, L, \delta, \Pi)$  is an *active tile type* if, for all  $d \in D$  and for all active glues  $(g, q), (g', q') \in G(d)$ ,  $g \neq g'$ . In other words, while a tile side may have multiple glues, there cannot be multiple copies of the same type of glue on a single side.

A transition function  $\delta$  takes as input a direction  $d \in D$  and a glue type  $g \in \Gamma$  (i.e. we say that it is “fired” by the binding of glue type  $g$  on side  $d$  of  $t$ ), and outputs a (possibly empty) set of *glue or label actions*, i.e., elements of  $(D \times \Gamma \times \{\text{on}, \text{off}\}) \cup (\Sigma \times \{\text{on}, \text{off}\})$ . Consider an active tile type  $t = (G, L, \delta, \Pi)$  and suppose that  $(g, q) \in G(d)$  for some  $d \in D$  and  $(d', g', q') \in \delta(d, g)$ . Intuitively, if  $m = \Pi(d', q')$  (i.e., the *multiplicity* of  $(d', q') \in \Pi$ ) before  $\delta$  “fires,” then  $\Pi(d', q') = m + 1$  after executing  $\delta$ . We assume for the sake of convenience that  $\delta$  outputs the empty set for any pair of direction and glue on which  $\delta$  is not explicitly defined. In other words, the binding of a glue on  $t$  fires the transition function, which can result in a set of “requests” for specific active glues and labels on  $t$  to transition into specified states.

As shorthand, let “ $-$ ” represent “**latent**”, “ $1$ ” represent “**on**”, and “ $0$ ” represent “**off**”. Let  $t = (G, L, \delta, \Pi)$  be an active tile type. For notational convenience, we will denote as  $g_{\{q|\exists(d,g,q)\in\Pi\}}^p$  the active glue  $(g, p)$  of  $t$  satisfying, for some  $d \in D$ ,  $(g, p) \in G(d)$ . We purposely omit the direction  $d$  from our shorthand notation because it will always be clear from the context. Note that, in our notation, the superscript specifies its current state (e.g. if the active glue is **on**, we write  $g^1$ ), and the subscript represents its set of *pending state transitions* (i.e., the set of all glue actions  $(d, g, q) \in \Pi$ ). For example, if  $\{(N, g, \text{on}), (N, g, \text{off}), (N, g, \text{off})\} \subseteq \Pi$ , then we write  $g_{1,0,0}^1$ . If there are no pending state transitions for  $g$  in  $\Pi$ , then we omit a subscript. Figure 2 shows the valid state transitions for active glues, which

is restricted to **latent** being able to transition to either **on** or **off**, and “**on**” being able to transition to “**off**”. Note that once a glue is in the **off** state, there is no valid transition out of that state. Also, the only valid initial states for an active glue are **latent** or **on** since a glue which started in the **off** state could never interact and therefore could simply be removed.

Figure 3 shows an example active tile type with  $G(N) = \{a^1, b^1\}$ ,  $G(W) = \{bl^-, br^-\}$ ,  $G(E) = \{bl^-, br^-\}$ ,  $G(S) = \emptyset$  and  $L = \{\text{Bot}\}$ . Note that for the sake of convenience, in figures we tend to omit superscripts unless the glue state is **off**, so a glue with no superscript and a rectangular tab next to it represents that the glue is **on**, and with no such a tab is **latent**. Glues which are **off** will contain the “0” superscript, or be removed completely, to remove ambiguity. Figure 3 also includes two depictions of the tile’s transition function. In this example, when the glue  $b$  on the North side of the tile binds, the tile’s transition function specifies that glues  $bl$  on the West side and  $br$  on the East side are requested to turn **on**.

An *active tile* is an instance of an active tile type which has its own pending sets for each of its active glues and labels—all of which sets must be initially empty.

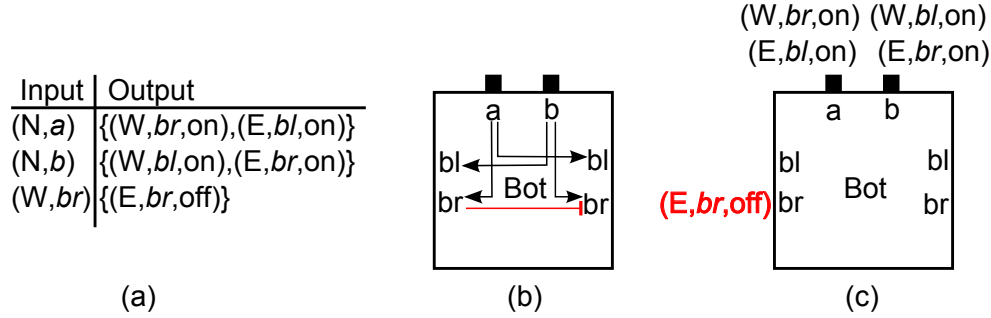


Fig. 3: An example active tile with (a) a tabular representation of its transition function, (b) a graphical depiction of its transition function using lines and arrows, and (c) a semi-graphical depiction. In (b), an arrowed line pointing from glue  $g_1$  to glue  $g_2$  denotes that the binding of  $g_1$  causes the glue action turning  $g_2$  on to be put into  $g_2$ ’s pending set  $P$ . A red line indicates that the action is to turn the destination glue off. In (c), the text descriptions of output actions for a given glue are placed next to its location on the tile edge. Note that for compactness we can also combine multiple actions. For example, if glue  $a$  also had output  $(W,bl,on)$  we could write  $(W,(bl,br),on)$ .

**Active Supertiles** Active supertiles in the STAM are defined similarly to supertiles in the 2HAM. Note that only glues which are in the **on** state can bind, and only those which are bound contribute to the  $\tau$ -stability of a supertile. When two  $\tau$ -stable supertiles can be translated so that they are non-overlapping and the abutting **on** glues can bind to create a cut strength of at least  $\tau$ , we say that they are  $\tau$ -combinable. A supertile  $A$  is said to be  $\tau$ -breakable into supertiles  $B$  and  $C$  if there exists a strength  $\tau' < \tau$  strength cut of the stability graph  $G_A$  that separates the tiles of  $A$  into supertiles  $B$  and  $C$ .

**Active Supertile Combination** Two active supertiles  $A$  and  $B$  may combine into active supertile  $C$  if the underlying supertiles for  $A$  and  $B$  are  $\tau$ -combinable into the underlying supertile for  $C$ . When supertiles combine, all matched glues in the **on** state on the boundary between  $A$  and  $B$  are said to bind, and thus fire the transition functions of their respective tiles, causing the necessary states to be added to the pending sets of the targeted tiles.

**Active Supertile Breaking** An active supertile  $A$  can break into active supertiles  $B$  and  $C$  if the underlying supertile for  $A$  has a cut of less than  $\tau$ -strength dividing  $A$  into the underlying supertiles for  $B$  and  $C$ .

### 3.1 Signal Passing Tile Assembly Model

An STAM system is a tuple  $(T, \tau)$  where  $T$  is a set of initial active supertiles referred to as the *initial assembly set* of the system, and  $\tau$  is a positive integer referred to as the *temperature* of the system. We further restrict that the initial assembly set only contains active super tiles  $t = (G, L, \delta, \Pi)$  such that  $\Pi = \emptyset$ . For some problems such as the unique assembly of shapes and lines (see section 4),  $T$  is further restricted to contain only active supertiles that consist of singleton active tiles.

**Producible Assemblies** The signal passing tile assembly model is defined in terms of the set of *producible* active supertiles  $P_{T,\tau}$ .  $P_{T,\tau}$  is defined recursively as follows:

**Base Set of Assemblies**  $T \subseteq P_{T,\tau}$

**Combination Transition** For any 2 active supertiles  $A, B \in P_{T,\tau}$ , if  $A$  and  $B$  are  $\tau$ -combinable into active supertile  $C$ , then  $C \in P_{T,\tau}$ .

**Break Transition** For any supertile  $A \in P_{T,\tau}$ , if  $A$  is  $\tau$ -breakable into active supertiles  $B$  and  $C$ , then  $B, C \in P_{T,\tau}$ .

**Active Glue Transition** Consider an active supertile  $A \in P_{T,\tau}$ . For each active tile  $t \in A$  with  $A(x, y) = t = (G, L, \delta, \Pi)$ , for all  $d \in D$ , for all  $(g, p) \in G(d)$  and for all  $(d, g, q) \in \Pi$ , there is a supertile  $B \in P_{T,\tau}$  such that  $A$  and  $B$  differ only at location  $(x, y)$  as follows:  $B(x, y) = t' = (G', L, \delta, \Pi')$  satisfying

1.  $\Pi' = \Pi - \{(d, g, q)\}$  and
2. if  $p = \text{on}$  and  $q = \text{off}$  or  $p = \text{latent}$  and  $q = \text{off}$  or  $p = \text{latent}$  and  $q = \text{on}$ , then for all  $d \neq d' \in D$ ,  $G'(d') = G(d')$ ,  $(g, q) \in G'(d)$  and for all  $q \neq q' \in Q$ ,  $(g, q') \notin G'(d)$ .

**Active Label Transition** Active label transitions are defined similar to active glue transitions.

**Terminal Assemblies** The set of producible assemblies of an STAM system defines the collection of active supertiles that can occur throughout the assembly process. The *terminal* set of assemblies  $\mathcal{A}_{\square}[T, \tau]$  of an STAM system  $(T, \tau)$  defines the subset of producible assemblies for which no combination, break, glue transition, or label transition is possible. Conceptually, the terminal assembly set represents the sink state of the assembly system in which the system has been given enough time for all assemblies to reach a terminal state. The terminal set of assemblies is considered the output of an STAM system. In our constructions we are interested in designing systems that will have either 1) a unique terminal assembly that has a desired shape or 2) has a collection of terminal assemblies in which the desired target assembly is the largest or 3) the desired target assembly is the only assembly which contains a specially designated *marker* tile.

Please see Section A.2 for an example STAM system and several transitions and producible assemblies.

### 3.2 Metrics

Within this paper we consider the problem of assembling precise shapes, simulating Turing machines, and the strict self-assembly of infinite fractals. To measure how efficiently we can solve these assembly problems within the STAM model, as opposed to alternate models, we consider a number of natural metrics designed to measure experimental feasibility with respect to likely STAM implementations, the most important being DNA origami based tiles which implement glue actions through DNA strand displacement.

**Tile Complexity** The tile complexity of an STAM system  $(T, \tau)$  is  $|T|$ , the number distinct active supertiles that serve as the initial set of assemblies for the system. In the case that  $T$  is restricted to contain active supertiles consisting of only singleton active tiles, this metric denotes the number of distinct tile types that must be engineered to implement the STAM system. In general, the metric describes the number of distinct assemblies that must be engineered to implement the system.

**Signal Complexity** The signal complexity of an STAM system is the maximum number of glues that occur on the face of any tile from  $T$ . Within the STAM system, each glue along a tile face is potentially set up to fire a transition function that triggers a set of local glue actions upon binding. In practice, such transition triggers can be set up through a sequence of DNA strand displacements. Setting up a small network of such displacement chains on the face of a DNA origami tile is experimentally quite feasible. However, the cost and complexity grow substantially as the number of distinct displacement reactions grows, making the signal complexity a key metric for experimental feasibility.

**Fuel Efficiency** Fuel efficiency is a metric that is considered in the context of simulating a Turing Machine and denotes the number of tiles that are used up (and cannot be reused) per computation step of the Turing machine being simulated. Our result constitutes the first Turing simulation self-assembly system that achieves  $O(1)$  fuel efficiency.

**Space Complexity** Space complexity is a metric that is considered in the context of simulating a Turing Machine and denotes the size of the assembly that represents the current tape and state of the Turing machine after a given computation step.

**Temperature** For an STAM system  $(T, \tau)$ , the temperature value  $\tau$  denotes the number of glue bonds required for two assemblies to combine. Higher temperature systems realize finer grained bonding strength differences than lower temperature systems and may be comparably harder to implement in practice. One of the most straightforward class of systems to implement may be temperature  $\tau = 1$  systems in which any single bond is sufficient to cause two assemblies to attach. In particular, strength  $\tau = 1$  systems do not require the implementation of error prone *cooperative bonding* in which attachment may be based on 2 or more glues spread across 2 or more distinct tiles.

## 4 Efficient Linear Assemblies

In this section we discuss the assembly of  $n \times 1$  lines. The first result is achieved with a recursive doubling strategy. Individual tiles join to form duples, which join to form length-4 assemblies, and so on. The second result uses glue deactivation to separate rectangles assembled using a counter into individual  $n \times 1$  lines.

**Theorem 1.** *There exists an STAM system  $(T, 1)$  which uniquely assembles a  $n \times 1$  line, for all  $n \in \mathbb{N}$ . Moreover,  $T$  contains 4 tiles, and uses  $O(\log n)$  glues per tile.*

**Theorem 2.** *There exists an STAM system  $(T, 1)$  which uniquely assembles a  $n \times 1$  line, for all  $n \in \mathbb{N}$ . Moreover,  $T$  contains  $O(1)$  tiles, and uses  $O(\frac{\log n}{\log \log n})$  glues per tile.*

See appendix Section A.3 for proof details of theorems 1 and 2.

## 5 Fuel Efficient Turing Machines

Showing that the original Tile Assembly Model is computationally universal is a simple matter of designing a tile assembly system which can simulate a universal Turing machine, as originally shown in [37], and later expanded upon in [7, 10, 13, 23]. While displaying the computational power of the TAM (and previous variants), a common drawback of the constructions has been the number of tiles utilized during the formation of the assembly which simulates the computation, which, in this paper, is referred to as the fuel efficiency of the simulation. For all prior constructions, it has been necessary to make a new copy of the entire tape of the Turing machine between each computational step, with the new copy identical to the original except for the slight difference of a mere two tape cells indicating 1. the output value in the tape cell left by the tape head, and 2. the tape cell marking the current location of the tape head. This full-scale copying of the tape, including the vast majority of cells which are unchanged, is wasteful in terms of the number of tiles required, experimentally very error prone due to the huge number of tile attachments required, and results in enormous assemblies. In this section, we exhibit a construction which is capable of simulating a universal Turing machine in the STAM, but while doing so only requires a small constant number of tiles (never more than 7) as fuel for each computational step and maintains an assembly which consists of a number of tiles which is only twice the total number of tape cells used by the Turing machine up to that step.

Throughout this paper, and without loss of generality, we define Turing machines as follows.

Let  $M$  be an arbitrary single-tape Turing machine, such that  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$  with state set  $Q$ , input alphabet  $\Sigma = \{0, 1\}$ , tape alphabet  $\Gamma = \{0, 1, \_ \}$ , transition function  $\delta$ , start state  $q_0 \in Q$ , accept state  $q_{\text{accept}} \in Q$ , and reject state  $q_{\text{reject}} \in Q$ . Furthermore,  $M$  begins in state  $q_0$  on the leftmost cell of the tape, expects a one-way infinite-to-the-right tape, and is guaranteed to never attempt to move left while on the leftmost tape cell.

**Theorem 3 (Fuel efficient Turing machines).** *For any Turing machine  $M$  with input  $w \in \{0, 1\}^*$ , there exists an STAM system  $\mathcal{T}_{M(w)} = (T_{M(w)}, 1)$  with tile complexity  $O(|Q|)$ , signal complexity  $O(1)$ , and fuel efficiency  $O(1)$ , which simulates  $M$  on  $w$  in the following way:*



1.  $T_{M(w)}$  contains an active supertile consisting of  $2|w| + 2$  active tiles representing  $w$  and  $M$ 's start state.
2. If  $M$  halts on  $w$ , then  $\mathcal{A}_{\square}[T_M, 1]$  contains exactly one supertile with  $> 3$  tiles and that supertile contains exactly one *ACCEPT* (*REJECT*) tile if  $M$  accepts (rejects)  $w$ .
3. If  $M$  does not halt on  $w$ , then  $\mathcal{A}_{\square}[T_M, 1]$  contains exactly 0 (terminal) supertiles with  $> 3$  tiles.

Our proof of Theorem 3 is by construction, the full details of which can be found in Section A.5 of the appendix. Here, we provide a brief overview.

Our construction works by utilizing a set of tile type templates that, along with the definition of a Turing machine  $M$ , are used to generate the set of active tiles which are specific to  $M$ . The construction uses a pair of tiles to represent each tape cell, with one tile representing the value (0, 1, or  $\_$ ) of that cell and one tile providing a “backbone” which the other attaches to and which also attaches to the backbone tiles of the cells to its left and right. Additionally, there is a special tile for the tape cell representing the rightmost end of the tape, and also - at any given time - exactly one tape cell which also represents one state of  $M$  along with the tape cell value. The location of the tape cell with that information denotes the location of  $M$ 's tape head at that point, and the value of the state tells what state  $M$  is in. Transitions of  $M$  occur in a series of 4 main steps in which tiles bind to the north of the tape cell denoting the head location, then to the north of the tape cell to the immediate left of right (depending on whether or not  $M$ 's transition function specifies a left or right moving transition from the current state while reading the current tape cell value), and along the way cause the dissociation of the tiles representing the tape cell values in both locations and their replacement with tiles which represent the correct output tape cell value of the transition and correctly record the new state and head location at the tape cell immediately to the left or right. Due to the asynchronous nature of glue deactivations, and also the necessity that any junk assemblies produced must not be able to attach to any portion of any supertile which represents any stage of the computation, junk assemblies are produced as size 2 or 3 so that any activated glues which may be able to bind to another supertile are hidden between the tiles composing the junk assembly. In such a way,  $M(w)$  is correctly simulated while requiring only a constant number of new tiles per simulated transition step, and all junk assemblies remain inert and at size either 2 or 3. If  $M(w)$  halts, there will be one unique, terminal supertile which represents the result of that computation and is of size  $> 3$ . If  $M(w)$  does not halt, only the junk assemblies will be terminal.

## 6 Self-Assembly of the Sierpinski Triangle

Discrete self-similar fractals are, as the name implies, the discrete version of self-similar fractals. As such, they are defined as sets of points in  $\mathbb{Z}^2$ , and they consist of infinite, aperiodic patterns. The fact that they are infinite and aperiodic makes it difficult, if not impossible, for them to strictly self-assemble in the aTAM, as is shown in [15, 24] where the impossibility of a class of discrete self-similar fractals, including the Sierpinski triangle, strictly self-assembling in the aTAM is proven. The Sierpinski triangle was also shown to be impossible to strictly self-assemble in the 2HAM in [4]. Additionally, Doty [12] has shown a generalization of the impossibility proof from [24] which applies to, among other things, scaled versions of the Sierpinski triangle for any scaling factor. Thus, any method of strictly self-assembling the Sierpinski triangle is of interest.

In this section, we show that weak self-assembly of the Sierpinski triangle is possible in the STAM with fewer tile types than existing aTAM constructions, and we also show that strict self-assembly at scale factor 2 is in fact possible in the STAM.

**The discrete Sierpinski triangle** (Here we use the definition of [15]) Let  $V = \{(1,0), (0,1)\}$ . Define the sets  $S_0, S_1, S_2, \dots \subset \mathbb{Z}^2$  by the recursion  $S_0 = \{(0,0)\}$ ,  $S_{i+1} = S_i \cup (S_i + 2^i V)$ , where  $A + cB = \{\mathbf{m} + \mathbf{cn} \mid \mathbf{m} \in A \text{ and } \mathbf{n} \in B\}$ . Then the (standard) discrete Sierpinski triangle is the set  $S_{\Delta} = \cup_{i=0}^{\infty} S_i$ . See Figure 4a for a depiction of the first five stages (i.e.  $S_0$  through  $S_4$ ).

### 6.1 Weak Self-Assembly of the Sierpinski Triangle

Here we describe an STAM system which weakly self-assembles  $S_{\Delta}$ . Of note is that this construction works at temperature 1, thus using no cooperative binding, and does not utilize glue deactivation.

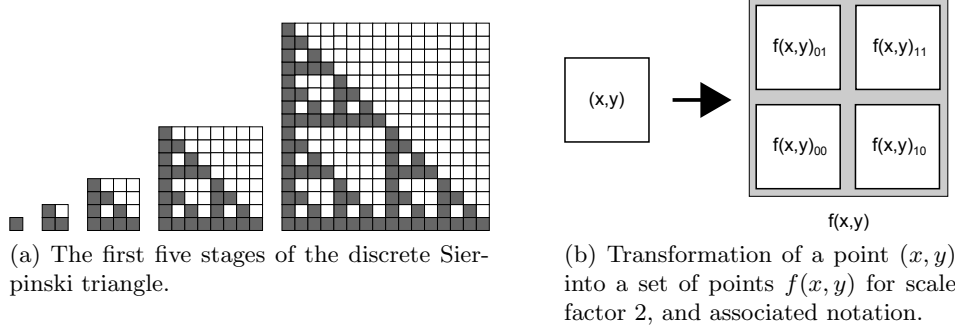


Fig. 4: The Sierpinski triangle and a description of the mapping for the scaled version.

Further, the tile complexity and signal complexity are small constants of 5 tiles and 4 signals per tile face. This construction is an example of a more general technique for simulating any  $\tau = 2$  rectilinear aTAM system (rectilinear systems grow from south to north, west to east) with a temperature  $\tau = 1$  STAM system. The general simulation of any  $\tau = 2$  aTAM system with a temperature  $\tau = 1$  system STAM system is direction for future work.

**Theorem 4.** *There exists an STAM system that weakly self-assembles the Sierpinski triangle. The system has 5 unique tiles, signal complexity = 4, assembles at temperature  $\tau = 1$ , and does not utilize glue deactivation.*

Our proof of Theorem 4 is by construction, the full details of which can be found in Section A.7 of the appendix. Here, we provide a brief overview.

The construction is very similar to standard aTAM constructions for weakly self-assembling the Sierpinski triangle (e.g. [27]), with the main difference being that that construction works at temperature  $\tau = 2$ . It essentially works by having each tile (which is not on an axis) attach with two input sides. Each input side receives as input either a 0 or a 1, and the value for both output sides is the **xor** of the input bits. Those tiles which output a 0 are colored white and considered outside of the Sierpinski triangle, and those which output a 1 are colored black and considered within it. Since our construction works at  $\tau = 1$ , one input direction - in this case the west - is chosen to be the first to bind. Then, signals cause glues for either possible value of the second input to be turned **on**, and whichever is able to bind is then able to activate the correct output glues and also turn **on** the correct label value which identifies the tile as being either white or black.

## 6.2 Strict Self-Assembly of the Sierpinski Triangle

**Theorem 5.** *There exists an STAM system that strictly self-assembles the discrete Sierpinski triangle, at temperature  $\tau = 1$ , tile complexity = 19, scale factor = 2, signal complexity = 5, and which makes use of glue deactivation, producing terminal junk assemblies of size  $\leq 6$ .*

Our proof of Theorem 5 is by construction, the full details of which can be found in Section A.8 of the appendix. Here, we provide a brief overview.

Define  $f(x,y)$  as the function which takes as input a point  $(x,y)$  and which returns the set of 4 points which correspond to  $(x,y)$  at a scale factor of 2, that is, the  $2 \times 2$  square of points  $\{(2x+a, 2y+b) \mid a, b \in \{0,1\}\}$ . (For instance,  $f(1,1) = \{(2,2), (2,3), (3,2), (3,3)\}$ ). For notation, we will refer to the 4 points in the set  $f(x,y)$  as  $f(x,y)_{00}$ ,  $f(x,y)_{01}$ ,  $f(x,y)_{10}$ , and  $f(x,y)_{11}$  as those corresponding to the values for  $a$  and  $b$  00, 01, 10, and 11, respectively. See Figure 4b for a clarification of this notation.

Let  $S_{2\Delta} = \{f(x,y) \mid (x,y) \in S_{\Delta}\}$  be the Sierpinski triangle at scale factor 2, i.e. where each point in the original Sierpinski triangle is replaced by a  $2 \times 2$  square of points, which we will refer to as a *block*. To prove Theorem 5, we now present an STAM system,  $\mathcal{T}_{2\Delta} = (T_{2\Delta}, 1)$  which strictly self-assembles  $S_{2\Delta}$ . At a high level, it does so by weakly self-assembling  $S_{2\Delta}$  by treating each block  $f(x,y)$  as a single tile which receives one input each from the block to its south and the block to its west. Each input is either a 0 or 1, and the block performs the equivalent of an **xor** operation on those inputs and outputs the result to its north and east. A block  $f(x,y)$  which outputs a 1 corresponds to a point  $(x,y) \in S_{\Delta}$  and thus a location which must remain tiled in the final assembly (shown as grey locations

in Figure 4a). A block  $f(x, y)$  which outputs a 0 instead corresponds to a point  $(x, y) \notin S_\Delta$  and which must eventually be removed from the final assembly (shown as white locations in Figure 4a). Whenever such a white region is completely tiled and also completely surrounded by blocks corresponding to grey positions (note that all white regions in  $S_\Delta$  are surrounded by grey positions), glue deactivation is used to “eject” the blocks of that white region as a set of “junk” supertiles. Those junk supertiles are then broken down into constant sized terminal supertiles (of sizes 3, 4, and 6) which are unable to attach to any portion of the infinitely growing assembly, and thus remain inert junk assemblies.

**Acknowledgments** The authors would like to thank Nataša Jonoska and Daria Karpenko for fruitful discussions and comments on this work.

## References

1. L. Adleman, Q. Cheng, A. Goel, MD Huang, D. Kempe, PM de Espanes, and PWK Rothmund, *Combinatorial optimization problems in self-assembly*, Proceedings of the thirty-fourth annual ACM symposium on Theory of computing, 2002, pp. 23–32.
2. G. Aggarwal, M. H. Goldwasser, M. Y. Kao, and R. T. Schweller, *Complexities for generalized models of self-assembly*, Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, 2004, p. 889.
3. F. Becker, *Pictures worth a thousand tiles, a geometrical programming language for self-assembly*, Theoretical Computer Science **410** (2009), no. 16, 1495–1515.
4. Sarah Cannon, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Matthew J. Patitz, Robert Schweller, Scott M. Summers, and Andrew Winslow, *Two hands are better than one (up to constant factors)*, Tech. Report 1201.1650, Computing Research Repository, 2012.
5. Ho-Lin Chen and David Doty, *Parallelism and time in hierarchical self-assembly*, Tech. Report 1104.5226, Computing Research Repository, 2011.
6. Qi Cheng, Gagan Aggarwal, Michael H. Goldwasser, Ming-Yang Kao, Robert T. Schweller, and Pablo Moisset de Espanés, *Complexities for generalized models of self-assembly*, SIAM Journal on Computing **34** (2005), 1493–1515.
7. Matthew Cook, Yunhui Fu, and Robert Schweller, *Temperature 1 self-assembly: Deterministic assembly in 3d and probabilistic assembly in 2d*, Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, 2011.
8. E.D. Demaine, M.L. Demaine, S.P. Fekete, M. Ishaque, E. Rafalin, R.T. Schweller, and D.L. Souvaine, *Staged self-assembly: nanomanufacture of arbitrary shapes with  $O(1)$  glues*, Natural Computing **7** (2008), no. 3, 347–370.
9. Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Mashhood Ishaque, Eynat Rafalin, Robert T. Schweller, and Diane L. Souvaine, *Staged self-assembly: nanomanufacture of arbitrary shapes with  $O(1)$  glues*, Natural Computing **7** (2008), no. 3, 347–370.
10. Erik D. Demaine, Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers, *Self-assembly of arbitrary shapes using rnase enzymes: Meeting the kolmogorov bound with small scale factor (extended abstract)*, Proceedings of the Twenty Eighth International Symposium on Theoretical Aspects of Computer Science (STACS 2011) (Dortmund, Germany), 2011, to appear.
11. R.M. Dirks and N.A. Pierce, *Triggered amplification by hybridization chain reaction*, Proceedings of the National Academy of Sciences of the United States of America **101** (2004), no. 43, 15275.
12. David Doty, personal communication, 2012.
13. David Doty, Lila Kari, and Benoît Masson, *Negative interactions in irreversible self-assembly*, DNA 16: Proceedings of The Sixteenth International Meeting on DNA Computing and Molecular Programming, Lecture Notes in Computer Science, Springer, 2010, pp. 37–48.
14. M.Y. Kao and R. Schweller, *Randomized self-assembly for approximate shapes*, Automata, Languages and Programming (2008), 370–384.
15. James I. Lathrop, Jack H. Lutz, and Scott M. Summers, *Strict self-assembly of discrete Sierpinski triangles*, Theoretical Computer Science **410** (2009), 384–405.
16. F. Liu, R. Sha, and N.C. Seeman, *Modifying the surface features of two-dimensional DNA crystals*, J. Am. Chem. Soc **121** (1999), no. 5, 917–922.

17. W. Liu, H. Zhong, R. Wang, and N.C. Seeman, *Crystalline Two-Dimensional DNA-Origami arrays*, *Angewandte Chemie International Edition* **50** (2011), no. 1, 264–267.
18. K. Lund, A.J. Manzo, N. Dabby, N. Michelotti, A. Johnson-Buck, J. Nangreave, S. Taylor, R. Pei, M.N. Stojanovic, and N.G. Walter, *Molecular robots guided by prescriptive landscapes*, *Nature* **465** (2010), no. 7295, 206–210.
19. U. Majumder, T.H. LaBean, and J.H. Reif, *Activatable tiles: Compact, robust programmable assembly and other applications*, *Proceedings of the 13th international conference on DNA computing*, 2007, pp. 15–25.
20. T. Omabegho, R. Sha, and N.C. Seeman, *A bipedal DNA brownian motor with coordinated legs*, *Science* **324** (2009), no. 5923, 67.
21. J.E. Padilla, W. Liu, and N.C. Seeman, *Hierarchical self assembly of patterns from the Robinson tilings: Dna tile design in an enhanced tile assembly model*, *Natural Computing* **online first**, **17 August 2011** (2011).
22. Jennifer Padilla, Wenyan Liu, and Nadrian Seeman, *Hierarchical self assembly of patterns from the robinson tilings: Dna tile design in an enhanced tile assembly model*, *Natural Computing* (2011), 1–16, 10.1007/s11047-011-9268-7.
23. Matthew J. Patitz and Scott M. Summers, *Self-assembly of decidable sets*, *Natural Computing*, to appear.
24. ———, *Self-assembly of discrete self-similar fractals*, *Natural Computing* **1** (2010), 135–172.
25. L. Qian and E. Winfree, *A simple DNA gate motif for synthesizing large-scale circuits*, *DNA Computing* (2009), 70–89.
26. Paul W. K. Rothemund and Erik Winfree, *The program-size complexity of self-assembled squares (extended abstract)*, *STOC '00: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing* (Portland, Oregon, United States), ACM, 2000, pp. 459–468.
27. Paul W.K. Rothemund, Nick Papadakis, and Erik Winfree, *Algorithmic self-assembly of DNA Sierpinski triangles*, *PLoS Biology* **2** (2004), no. 12, 2041–2053.
28. P.W.K. Rothemund, *Folding DNA to create nanoscale shapes and patterns*, *Nature* **440** (2006), no. 7082, 297–302.
29. P.W.K. Rothemund, N. Papadakis, and E. Winfree, *Algorithmic self-assembly of DNA sierpinski triangles*, *PLoS Biology* **2** (2004), no. 12, e424.
30. P.W.K. Rothemund and E. Winfree, *The program-size complexity of self-assembled squares (extended abstract)*, *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000, pp. 459–468.
31. G. Seelig, D. Soloveichik, D.Y. Zhang, and E. Winfree, *Enzyme-free nucleic acid logic circuits*, *science* **314** (2006), no. 5805, 1585.
32. D. Soloveichik and E. Winfree, *Complexity of self-assembled shapes*, *SIAM Journal on Computing* **36** (2007), no. 6, 1544–1569.
33. H. Wang, *Proving theorems by pattern recognition II*, *AT&T Bell Labs Tech. J* **40** (1961), 1–41.
34. S.F.J. Wickham, M. Endo, Y. Katsuda, K. Hidaka, J. Bath, H. Sugiyama, and A.J. Turberfield, *Direct observation of stepwise movement of a synthetic molecular transporter*, *Nature Nanotechnology* **6** (2011), no. 3, 166–169.
35. E. Winfree, *Algorithmic self-assembly of dna*, Ph.D. thesis, California Institute of Technology, May 1998.
36. E. Winfree, F. Liu, L.A. Wenzler, N.C. Seeman, et al., *Design and self-assembly of two-dimensional DNA crystals*, *Nature* **394** (1998), no. 6693, 539–544.
37. Erik Winfree, *Algorithmic self-assembly of DNA*, Ph.D. thesis, California Institute of Technology, June 1998.
38. P. Yin, H.M.T. Choi, C.R. Calvert, and N.A. Pierce, *Programming biomolecular self-assembly pathways*, *Nature* **451** (2008), no. 7176, 318–322.
39. B. Yurke, A.J. Turberfield, A.P. Mills, F.C. Simmel, and J.L. Neumann, *A DNA-fuelled molecular machine made of DNA*, *Nature* **406** (2000), no. 6796, 605–608.
40. D.Y. Zhang and G. Seelig, *Dynamic DNA nanotechnology using strand-displacement reactions*, *Nature Chemistry* **3** (2011), no. 2, 103–113.

## A Appendix

### A.1 Section 2: Physical Basis for the Model

DNA strand displacement reactions proceed via a branch migration resembling a random walk at a rate that can vary widely depending on toehold length and composition [40]. We allow for the time taken by signaling or glue activation events carried out by DNA strand displacement by storing actions triggered by various events in STAM assembly in a queue for asynchronous processing. STAM assembly retains the asynchronous nature of TAM assembly, thus there must be a way to allow for triggered events to happen in various orders in the model. We consider a DNA strand displacement mechanism that may serve as a basis for triggering the signaling process of the STAM. It begins with a toehold mediated binding event that brings two tile edges together, Figure 5. The strand displaced by binding can now serve to initiate some other event that begins a signal cascade across the tile.

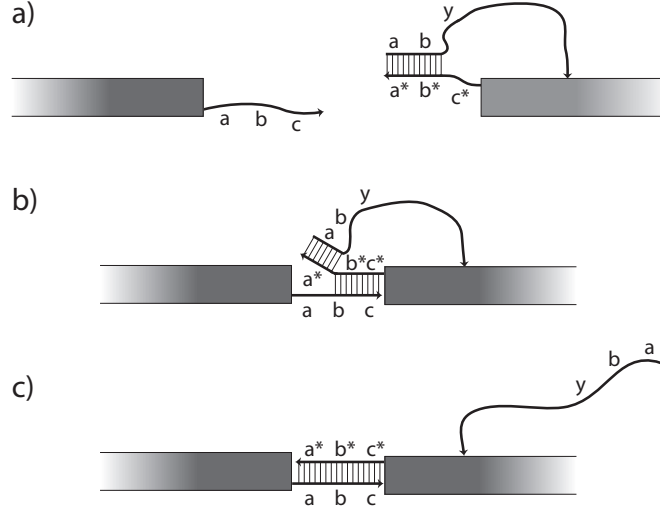


Fig. 5: DNA mechanism for tile binding and triggering of a signal cascade. a) Complementary DNA strands of active glues can initiate binding at the toehold  $c^*$ . b) Toehold binding initiates a 3-strand DNA branch migration in which the  $abc$  strand displaces the  $yba$  strand. c) Tiles binding is complete and the displaced strand is now free to trigger a signal cascade. In particular, the DNA sequence labeled  $y$  that has been made available to trigger the signal cascade is different from the binding sequence  $abc$ , so that the binding event itself, and not a random collision with the  $abc$  sticky end causes the signal cascade to be triggered.

### A.2 Section 3: STAM Notation and Model

**Example STAM System** Here we give a small example STAM system along with descriptions of several transitions and producible assemblies.

**Example tile types** Figure 6 shows a tile set that will be used for the following example. We will refer to active tiles by their labels for convenience. The “Bot” tile has four glue actions defined for its transition function, all of which turn **on** glues. The “MM” tile includes two glue actions which turn **off** glues: when glue  $e$  on the North binds, both that glue and glue  $b$  on the South get **off** transitions placed into the  $\Pi$  multiset of the active tile, which will eventually cause them to be turned **off**. The transition function for the “MR” tile contains a label action, namely when the  $c$  glue on the South binds.

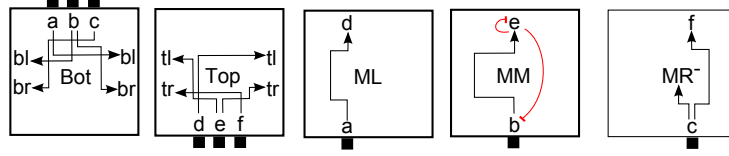


Fig. 6: An example active tile set.

**Example STAM transitions** Define an STAM system  $(T, 2)$  where  $T$  is the tile set shown in Figure 6. Figure 7 shows a set of transitions and producible assemblies within that system. Figure 7(a) shows a “Top” and “MM” tile in their initial configurations (with empty  $\Pi$  sets for all active tiles as required, and all superscripts explicitly shown for reference). Figure 7(b) shows the supertile resulting from a combination transition performed on one “Top” and one “MM” tile. Note that the binding of the two  $e$  glues causes four glues to have state transitions added to the  $\Pi$  multisets they belong to their respective active tile. Figure 7(c) shows the supertile resulting from a glue flip transition performed on the supertile in Figure 7(b), namely the  $tl$  glue on the West side of the “Top” tile transitioned to state **on** and that pending transition was removed from the  $\Pi$  multiset belonging to its active tile. Figure 7(d) shows the supertile resulting after two glue flip transitions are performed on the supertile in Figure 7(c), namely glue  $tr$  on the East of “Top” is turned **on** and  $e$  on the North of “MM” is turned **off**.

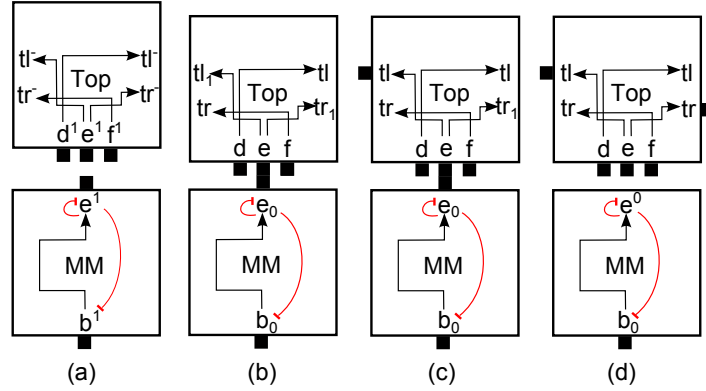


Fig. 7: Example transitions for the tile types in Figure 6 assuming a temperature  $\tau = 1$  system

### A.3 Section 4: Efficient Linear Assemblies

We first state our result for the special case where  $n$  is a power of 2 and provide a detailed construction. The exact tile set is given in Figure 11. A high level description of the recursive doubling technique is given in Figure 10, and a small example of the assembly process for a length 8 line is given in Figure 12. This result can be generalized to work for any positive integer  $n$  by increasing the glues on the tile types of the powers of 2 construction, yielding the final result stated in Theorem 1.

**Theorem 6.** *There exists an STAM system  $(T, 1)$  which uniquely assembles a  $2^k \times 1$  line, for all  $k \in \mathbb{N}$ . Moreover,  $T$  contains 4 tiles, and uses  $O(k)$  glues per tile.*

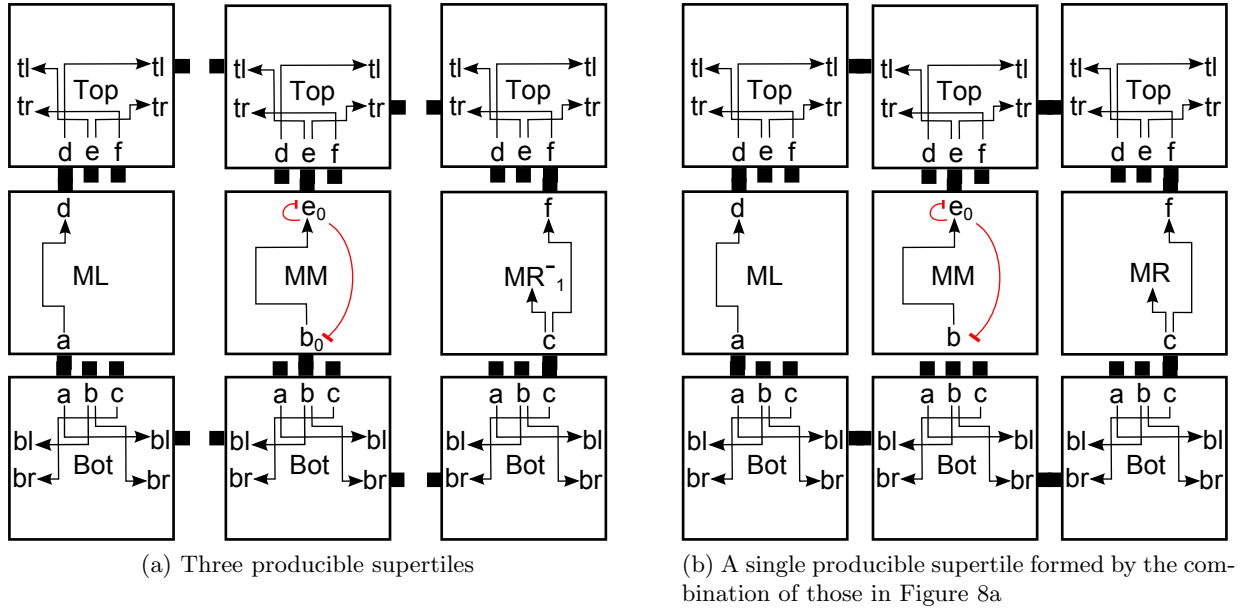


Fig. 8: Productible supertiles in the temperature  $\tau = 1$  system defined with the tile types in Figure 6

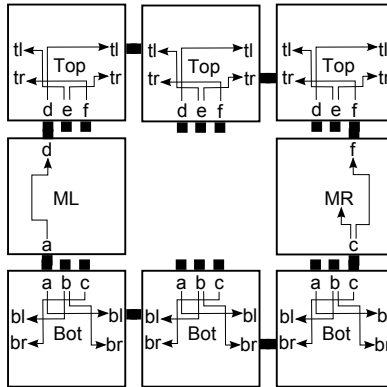


Fig. 9: Another producible supertile in the temperature  $\tau = 1$  system defined with tile types from Figure 6. Note that it is not terminal as a new “MM” tile could attach in the center, repeatedly falling off and being replaced an infinite number of times.

## Proof of Theorem 6

*Proof.* Define  $f(A)$  as a naming function that takes as input supertile  $A$  and returns a string  $\in \{a_k, b_k, x_k, y_k \mid k \in \mathbb{N}\}$ . For an arbitrary input  $A$ , if the height is 1, and the width is  $2^n$  for  $n \in \mathbb{N}$ ;  $f(A)$  returns  $a_k$  if  $A$  has active glues labeled  $ax_k$  and  $ay_k$  on its eastern edge,  $b_k$  if  $A$  has active glues labeled  $bx_k$  and  $by_k$  on its eastern edge,  $x_k$  if  $A$  has active glues labeled  $ax_k$  and  $bx_k$  on its western edge, and  $y_k$  if  $A$  has active glues labeled  $ay_k$  and  $by_k$  on its western edge. Otherwise,  $f(A)$  returns  $\emptyset$ . Initially, tileset  $T$  consists of four singleton tiles whose names correspond to their identities as defined by  $f(A)$ :  $a_0, b_0, x_0$ , and  $y_0$ . Assembly proceeds by recursive combination of these tiles into larger supertiles. Figure 10 shows how producible assemblies can join; for example,  $a_1$  can attach to the western edge of either  $x_1$  or  $y_1$ , producing  $a_2$  and  $y_2$ , respectively. For a detailed view of the internal signal structure of the tiles that allows for this behavior, see figure 11. Note that each assembly possesses two exposed external glues, allowing non-deterministic binding to occur, a critical aspect of the construction.

This binding event that occurs as a result of joining two assemblies together will *propagate* a signal either left or right, depending on what identity the newly formed assembly is supposed to take on. For example: if tile  $a_0$  attaches to tile  $x_0$ , it can only do so by virtue of glue  $ax_0$ . At this point, the assembly "knows" that it is now supposed to be  $a_1$ , and activates a glue on its right face:  $R_1$ . Each tile in the assembly can receive  $R_1$  on its left edge (because it is already activated), and activate it on its right edge, effectively propagating a signal across multiple tiles. When the rightmost tile (in this case,  $x_0$ ) receives the signal, it activates glues corresponding to the appropriate identity, i.e.,  $ax_1$  and  $ay_1$ . At this point, the assembly can properly be called  $a_1$ . The tileset is designed in such a way that the external edge tiles in any particular assembly are known. For example, it can be observed that the leftmost and rightmost tiles in any  $a_k$  will be  $a_0$  and  $x_0$ , respectively. This fact allows a simple signal transduction: when an edge tile receives a particular signal, there is exactly one identity that particular assembly can adopt via glue activation.

The assembly process can be thought of as being isomorphic to a staged assembly process, where at every stage  $k$  such that  $0 < k \leq \log_2 n$ , four new assemblies are formed from assemblies produced at stage  $k-1$ . These can combine with one another in four ways, and so on. This process terminates at the final mixing, when  $k = \log n - 1$ , because the final binding event does not trigger signal propagation, resulting in a terminal  $n \times 1$  line.

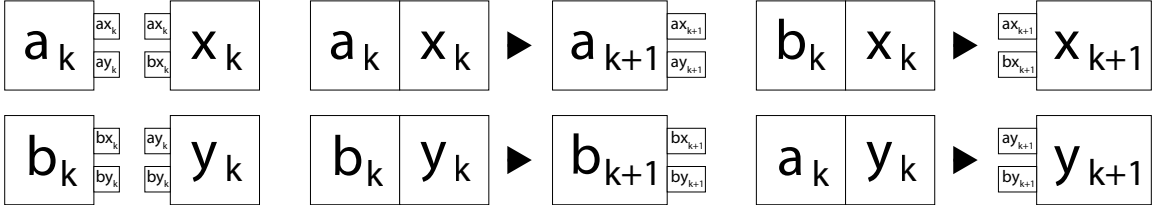


Fig. 10: Four recursively defined types of assemblies:  $a_k, b_k, x_k, y_k$ . For example:  $a_k$  is composed of  $a_{k-1}$  and  $x_{k-1}$  and has glues  $ax_k$  and  $ay_k$  activated on its eastern edge.

### A.4 Proof of Theorem 1

*Proof.* Let  $n \in \mathbb{N}$  be an arbitrary integer. Then we may write  $n$  as a sum of powers of 2,  $n = \sum_{i \in I} 2^{j_i}$ , where  $k = \lfloor \log n \rfloor$ ,  $J = \{j_1, \dots, j_{m-1}, j_m = k\} \subseteq \{0, 1, \dots, k\}$ ,  $I = \{0, 1, \dots, |J| = m\}$ . Just as the number  $n$  can be written this way, the line of length  $n$  can be composed of segments with power of 2 lengths, each composed during the construction of the line of length  $2^k$ , as described in the previous section. The final joining of line segments corresponding to the powers of 2 that sum to  $n$  is shown in Figure 13.



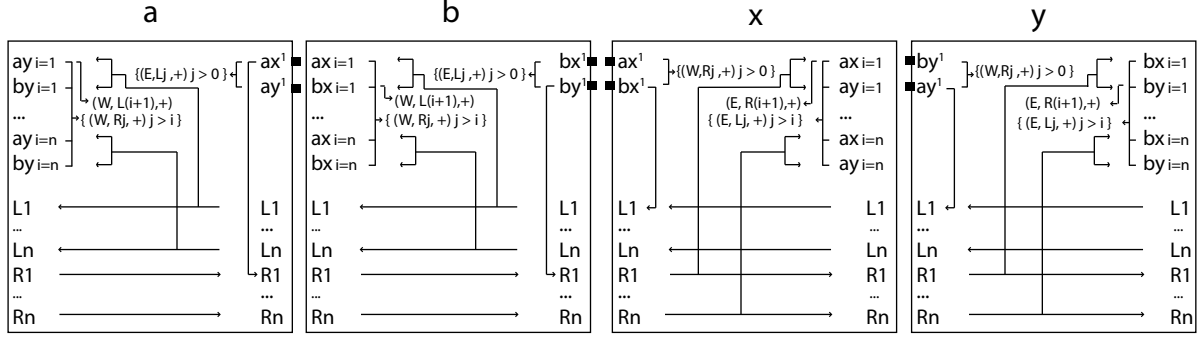


Fig. 11: The tileset for producing a  $1 \times n$  line, where  $n$  is a power of 2.

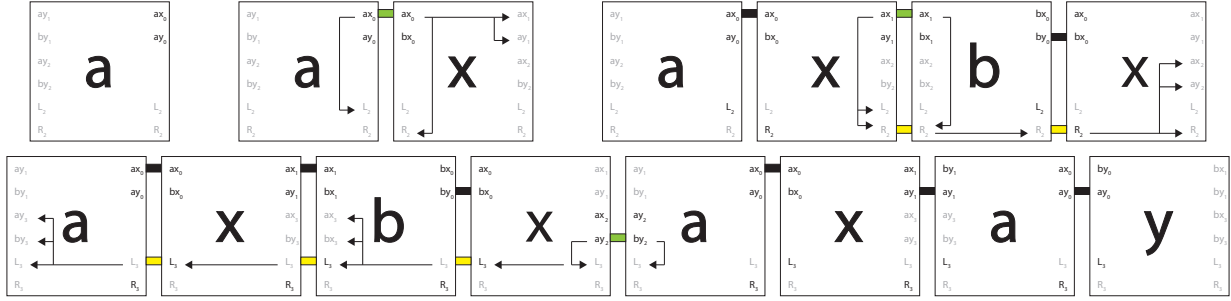


Fig. 12: Example assembly of length 8 line.

The tiles from the previous section are modified to include up to  $k$  more glues each that direct the last stage of assembly shown in Figure 13. The resulting 4 tiles still have  $O(\log n)$  glues per tile. The modified tiles are shown in Figure 14.

**Proof of Theorem 2** In this section we provide a construction to prove Theorem 2 which states that there exists an STAM system  $(T, 1)$  which uniquely assembles a  $n \times 1$  line, for all  $n \in \mathbb{N}$ , utilizing  $O(1)$  tiles, and  $O(\frac{\log n}{\log \log n})$  glues per tile.

The construction is based on temperature  $\tau = 2$  counters that have been developed within the ATAM and 2HAM. Our approach is to modify one of these constructions to efficiently build a  $k \times n$  rectangle with  $O(\log n / \log \log n)$  tile types but at temperature  $\tau = 1$  through the use of signalling. The tile set for a base  $m = n^{1/k}$ ,  $k$ -digit counter tile system for the assembly of a  $k \times n$  rectangle is described in Figure 15. The construction grows from an initial length  $k$  seed row whose exposed north glues denote the starting value of the counter, and in turn the length of the final assembly. Growth proceeds northward with the property that only the northmost row that contains placed tiles contains fewer than  $k$  placed tiles. As tiles begin to bind to the north face of a seed assembly the tile placed on the rightmost column is incremented from its seeded value, while the tile placed in the leftmost column

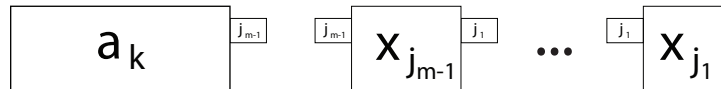


Fig. 13: Last step of line segment assembly to produce line of length  $n$ . Supertiles here are composed just as in Figure 10

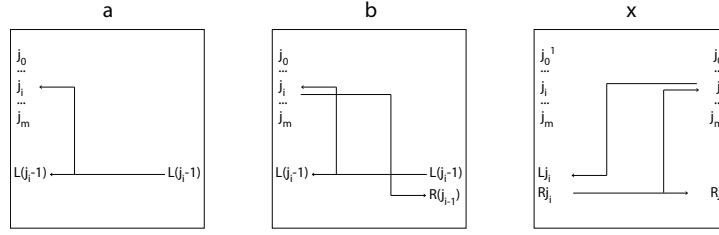


Fig. 14: Tiles modified to produce line of length  $n$ . All tiles retain all glues and signals from Figure 11 and add the glues and signals shown here. On tile x, glue  $j_0$  is active only if  $j_0 = 1$ .

base m counter, for  $i = 0$  to  $m-2$ ,  $j = 1$  to  $k-1$

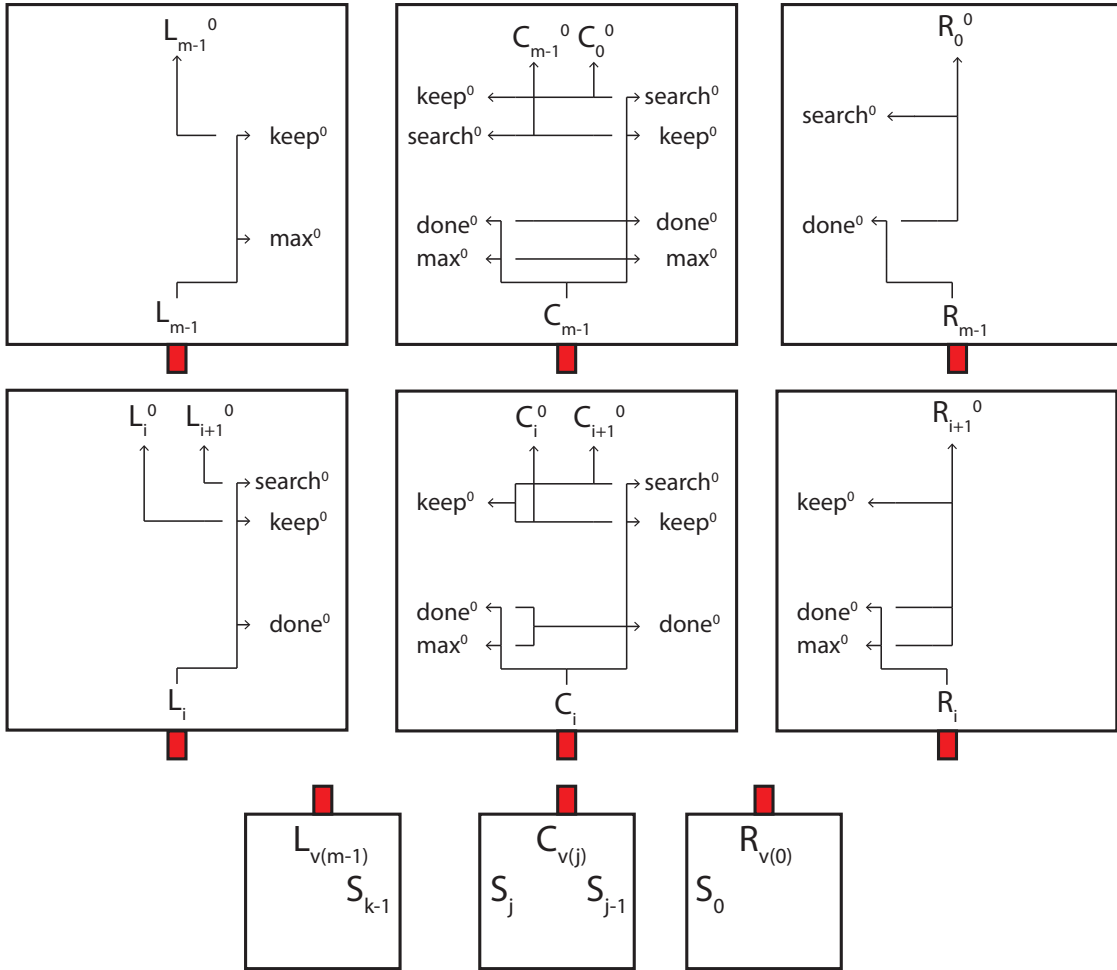


Fig. 15: tileset for base m k-digit counter, for  $i = 0$  to  $m-2$  and  $j = 1$  to  $k-1$ .

will begin to propagate a signal east. When the signal reaches the rightmost column, the current row is finished assembling. A second signal is then propagated west, which exposes a glue representing 0 on the north face of each tile until it encounters a tile valued at something less than  $m - 1$ . At that point, it increments the current value, and proceeds to propagate the signal, keeping the values rather than resetting them. This entire process continues until the east-to-west signal encounters  $m - 1$  at every position along the counter, indicating that it is maxed out, at which point the rectangle assembly halts. Figure 16 depicts the steps that occur each time the counter is incremented. The base-9 counter is initially set to 787. Signal  $a$  propagates east, ensuring that the entire row is in place. Upon reaching the eastmost (least significant) value, it initiates signal  $b$ . This propagates west, until it encounters a value less than 8. It immediately encounters 7, which is incremented. All more significant bits retain their value, and the counter is incremented to 788. In the next level of execution, signal  $d$  performs the same function as  $a$ . Signal  $e$  is propagated west until it encounters a value less than 8. Each value before this point is reset to 0, and the counter is incremented to 800.

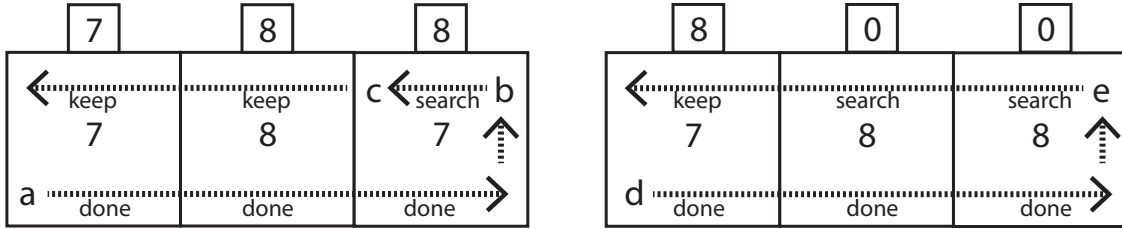


Fig. 16: base-8 counter

In order to encode arbitrary values of  $n$  for the length of the rectangle, the glues on the northern faces of the seed tile are set to encode the integer  $p$ , such that  $m^k - p = n$ . We now discuss how to extend our temperature  $\tau = 1$  counter construction into a line building construction by utilizing glue deactivation to break assembled length  $n$  rectangles up into  $1 \times n$  lines.

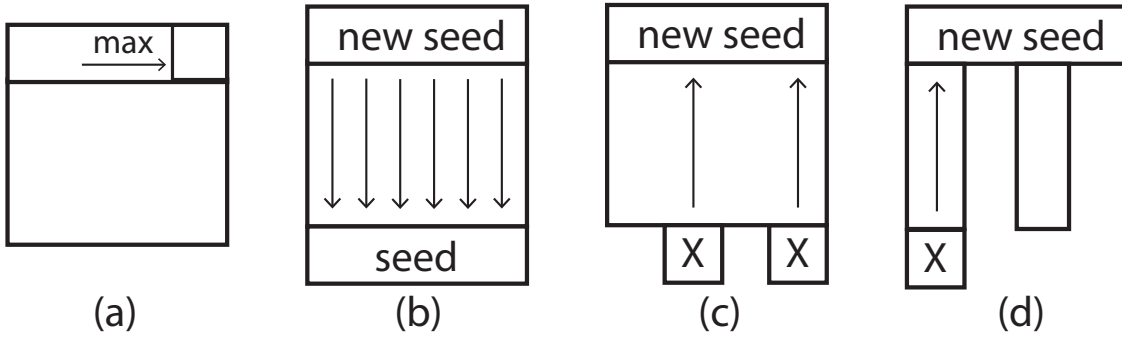


Fig. 17: A  $k \times m$  rectangle is broken into  $1 \times m$  lines

Figure 17 depicts the process. When the counter maxes itself out, a tile labeled “max” reaches the rightmost column of the counter as shown in Figure 17 (a). The placement of this final “max” tile initiates a chain of signals, shown in Figure 17 (b), that travel south down each column of the counter and turn off the south glues connecting the initial seed row to the counter assembly, as well as turn on north glues for the top row exposing a pattern that represents the initial seed value of the original

counter. The tiles of the original seed row may then detach, exposing a newly activated  $x$  glue which permits the attachment of a tile  $x$  shown in Figure 17 (c). The attachment of tile  $x$  initiates a chain of signals that travel north through the counter column, queuing up commands to turn off all east/west glues, as well the northmost glue. With these glues turned off, the length  $n$  lines are free to detach as shown in Figure 17 (d). The newly assembled seed also detaches to initiate a new counter.

It is possible that detached  $1 \times n$  columns will still have exposed glues in the on state and could attach to one another as shown in Figure 18 (a). However, as all glues that would yield such attachment are queued to be shut off, such erroneous assemblies are not terminal. Additionally, we must ensure that detached  $1 \times n$  columns cannot attach to not-yet-built counter assemblies, which could potentially mess up the counter assembly. However, as each column is capped on its south by a tile  $x$ , and each growing counter row is full except for the northern most row (Figure 18 (b)), the geometry prevents detached columns from lining up equal glues to cause an erroneous attachment.

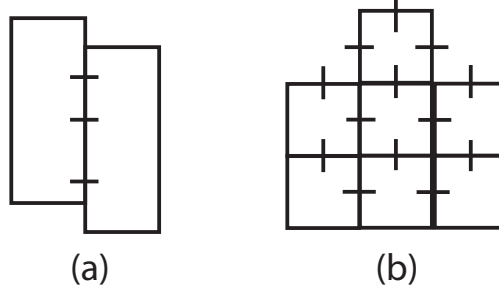


Fig. 18: Undesired assembly

## A.5 Section 5: Fuel Efficient Turing Machines

Throughout this paper, and without loss of generality, we define Turing machines as follows.

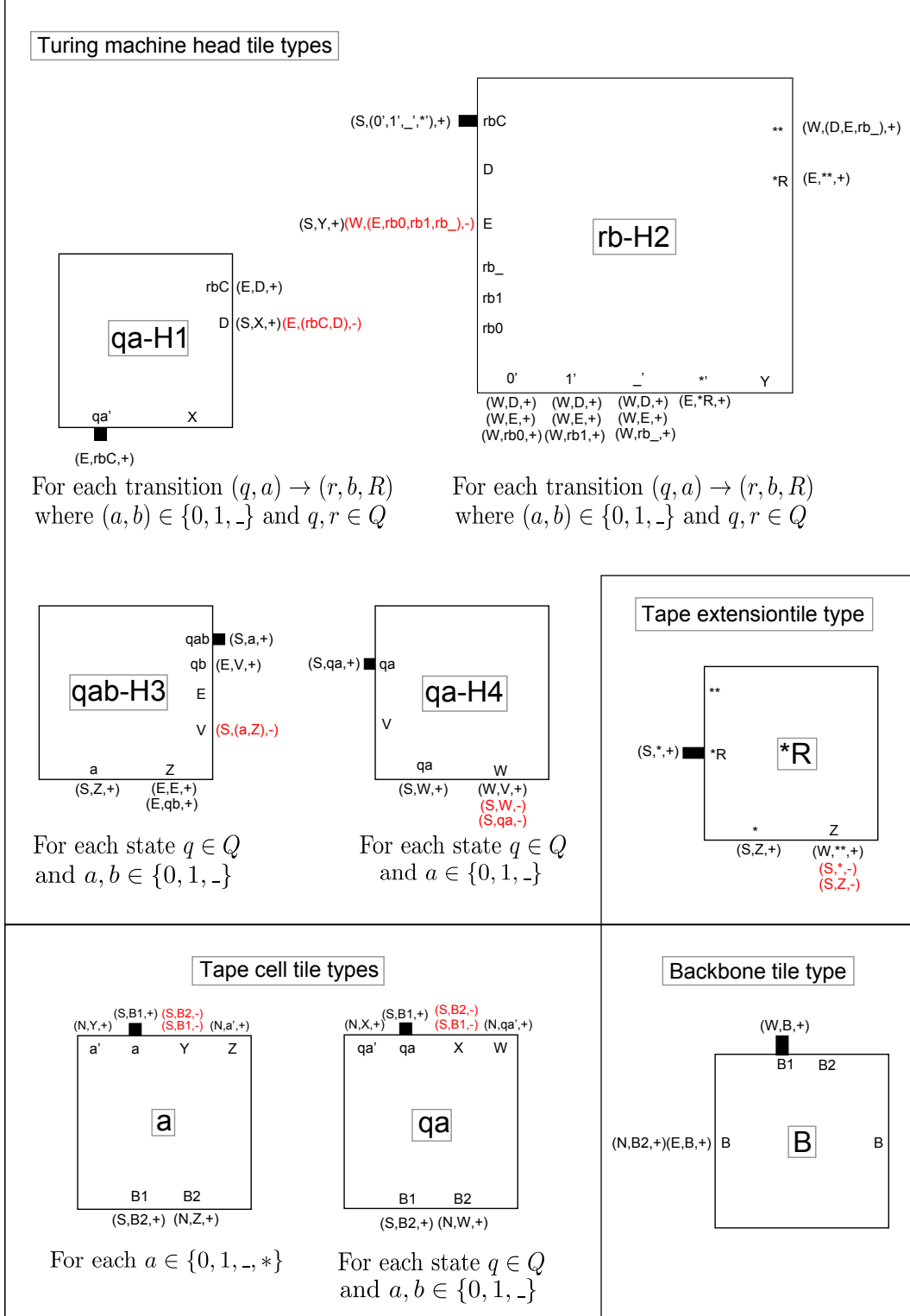
Let  $M$  be an arbitrary single-tape Turing machine, such that  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$  with state set  $Q$ , input alphabet  $\Sigma = \{0, 1\}$ , tape alphabet  $\Gamma = \{0, 1, \_ \}$ , transition function  $\delta$ , start state  $q_0 \in Q$ , accept state  $q_{\text{accept}} \in Q$ , and reject state  $q_{\text{reject}} \in Q$ . Furthermore,  $M$  begins in state  $q_0$  on the leftmost cell of the tape, expects a one-way infinite-to-the-right tape, and is guaranteed to never attempt to move left while on the leftmost tape cell.

**Theorem 3 (Fuel efficient Turing machines).** *For any Turing machine  $M$  with input  $w \in \{0, 1\}^*$ , there exists an STAM system  $\mathcal{T}_{M(w)} = (T_{M(w)}, 1)$  with tile complexity  $O(|Q|)$ , signal complexity  $O(1)$ , and fuel efficiency  $O(1)$ , which simulates  $M$  on  $w$  in the following way:*

1.  $T_{M(w)}$  contains an active supertile consisting of  $2|w| + 2$  active tiles representing  $w$  and  $M$ 's start state.
2. If  $M$  halts on  $w$ , then  $\mathcal{A}_{\square}[T_M, 1]$  contains exactly one supertile with  $> 3$  tiles and that supertile contains exactly one *ACCEPT* (*REJECT*) tile if  $M$  accepts (rejects)  $w$ .
3. If  $M$  does not halt on  $w$ , then  $\mathcal{A}_{\square}[T_M, 1]$  contains exactly 0 (terminal) supertiles with  $> 3$  tiles.

*Proof (Proof sketch).*

Our proof is by construction. Let  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$  be a Turing machine and  $w \in \{0, 1\}^*$  be the input. We define the finite set of tile types  $T_{M(w)}$  in relation to  $M$  as shown in Figure 19. Note that the tile types presented are generic templates applicable for generating the tile types specific to a given  $M$  (with the exception of the “tape extension tile type”, “backbone tile type”, and the left “tape cell tile type” which are used for all  $M$ ). Also note that the tile type templates provided for the “Turing machine head tile types” are specific to transitions which move the tape head to the right, and those for left-moving transitions are simply mirror images with a unique set of glues (e.g. every



glue is prefixed with “ $L$ ”) defined specifically for those transitions. The actual tile types that would be generated for a specific  $M$  can be determined by substituting each valid variable value (as specified below each tile type template) into glue labels, and potentially consist of a large number of actual tile types generated by each template definition.

The tiles in the “Tape cell tile types” and “Backbone tile type” sections of Figure 19 are used to compose the tape. The tape itself, as shown in Figure 20, consists of a row of east-west connected “backbone” tiles, each of which is connected on its north to a tile representing a tape cell. The backbone is used to keep the entire assembly connected during the process of tape cell replacement. The possible values for tape cells are: (1) “0”, (2) “1”, (3) “\_” (a blank), (4) “\*” (which represents the currently rightmost tape cell), or (5)  $q \times \{0, 1, \_ \}$  where  $q \in Q$  (at any point there is exactly one such tape cell which represents the location of the head and  $M$ ’s current state along with the contents of the tape cell that the head is currently reading. The tiles in the “Turing machine head tile types” group are used to simulate the actions of the head, which can be understood as four high level steps ( $H1, H2, H3$ , and  $H4$  - to be explained shortly). The “Tape extension tile type” is used to extend the tape by one additional blank symbol (“\_”) if and when the head reaches the rightmost end of the tape, thus allowing the assembly to simulate a one-way infinite-to-the-right tape.

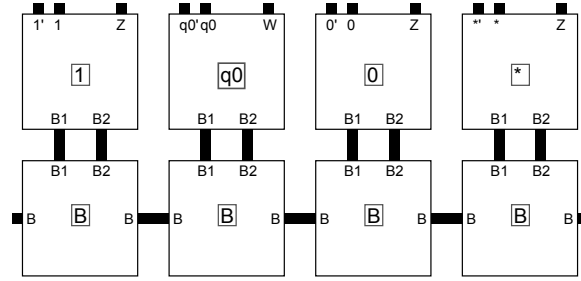


Fig. 20: Start state of the Turing machine  $M$  simulation example assuming a tape of “100\*” with  $M$  in state  $q$  and currently reading the leftmost 0.

The easiest method of explanation of this construction is to step through an example transition, which we now do. The specific example is depicted in the series of Figures 20, 21 and 22. Assume that  $M$  is currently in state “ $q$ ” and reading symbol “0”, the output symbol is “1”, the next state should be “ $r$ ”, and the head should move right (i.e. the simulated transition is  $(q, 0) \rightarrow (r, 1, R)$ ). Glues which are currently **on** are shown as black, glues which have just been turned **on** or connected in the current step are shown as bright blue, glues which are queued to turn **off** are shown as red, and glues which are currently **off** or **latent** have been hidden. Triggered actions have been depicted by arrows.

The transition begins from the  $q0'$  glue on the  $q0$  tile. As depicted in Figure 21(a), the  $q0-H1$  tile encodes the information to read a 0 while in state  $q$ , change it to a 1, move the head to the right, and change to state  $r$ . The values  $r$ , 1 and “move to the right” are implicitly stored by  $r1-H2$  as shown in Figure 21(b).  $r1-H2$  then connects to the tape cell tile below to read the symbol on that cell (a 0 here), and sends a message back to  $q0-H1$  which causes  $q0-H1$  and the now obsolete tape cell beneath it to dissociate as a pair, as shown in Figure 21(c)-Figure 21(f). When  $r1-H2$  connected with  $q0-H1$ ,  $r1-H2$  didn’t know what the tape cell value beneath it was, so it had to activate all four possible glues, which are  $0'$ ,  $1'$ ,  $\_'$ , and  $*'$ . Since only the  $0'$  binds,  $r1-H2$  can pass along the value of 0 in its next glue  $r10$  to a tile of type  $r10-H3$ . Here the  $r10$  means the next state will be  $r$ , the tape cell of the current head position will change to 1, and the symbol of next head position is 0. Then, in Figure 21(h)-Figure 21(l), the  $r10-H3$  tile causes a new tile representing a tape cell value of 1 to fully bind to the backbone and activate its  $1'$  glue, which completes the transition of that tape cell. After that 1 tile is connected to the backbone, it sends a message to  $r1-H2$  that allows it to dissociate, as shown in Figure 21(l)-Figure 21(o). Finally, the  $r0-H4$  tile binds to  $r10-H3$ ’s  $r0$  glue, allowing it to facilitate the binding of the  $r0$  where the 0 previously was, and eventually dissociate along with the  $r10-H3$  once the  $r0$  is fully connected, as shown in Figure 21(p)-Figure 22(w). At this point, the tape

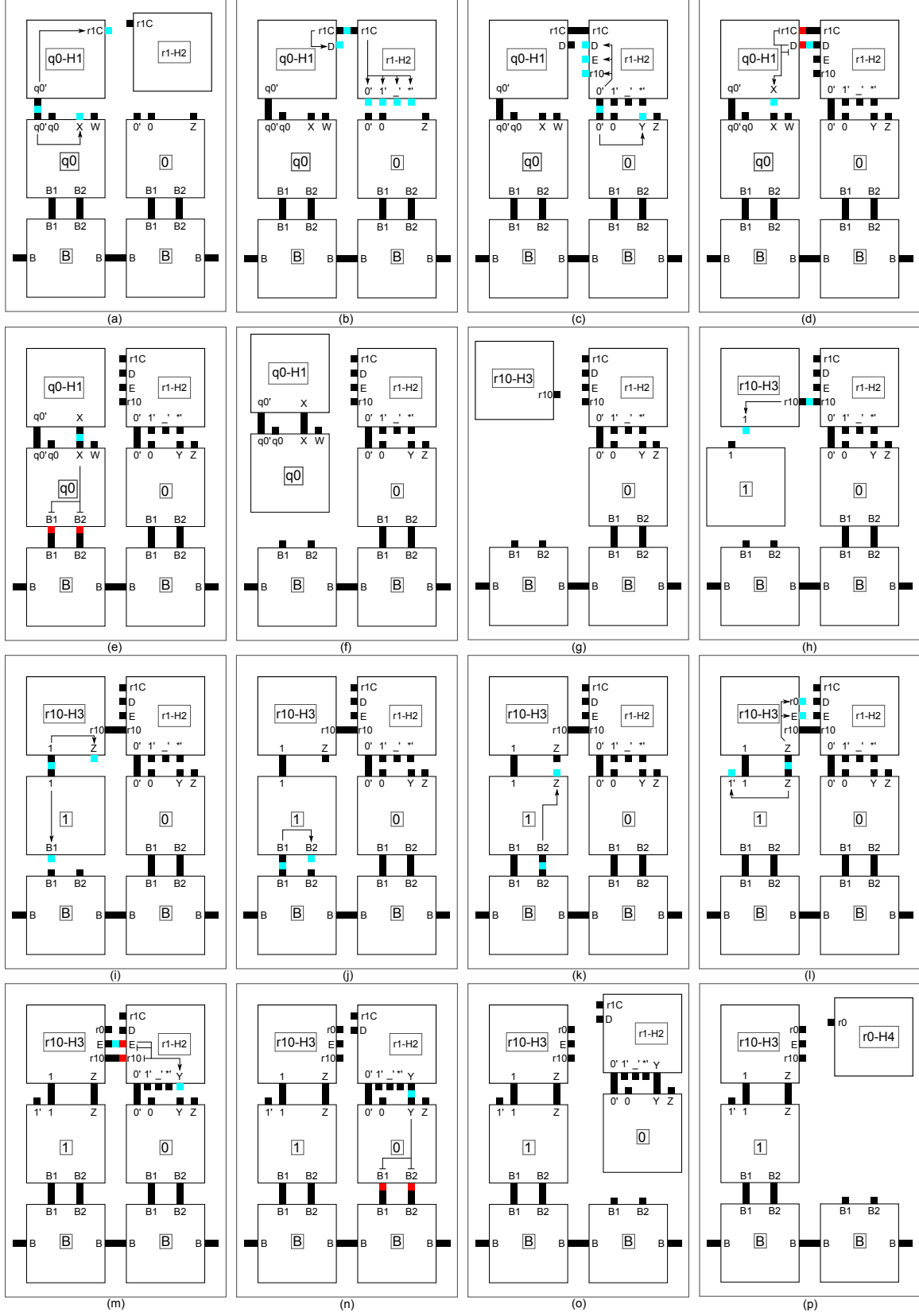


Fig. 21: High-level sketch of the simulation of a transition, part 1 of 2

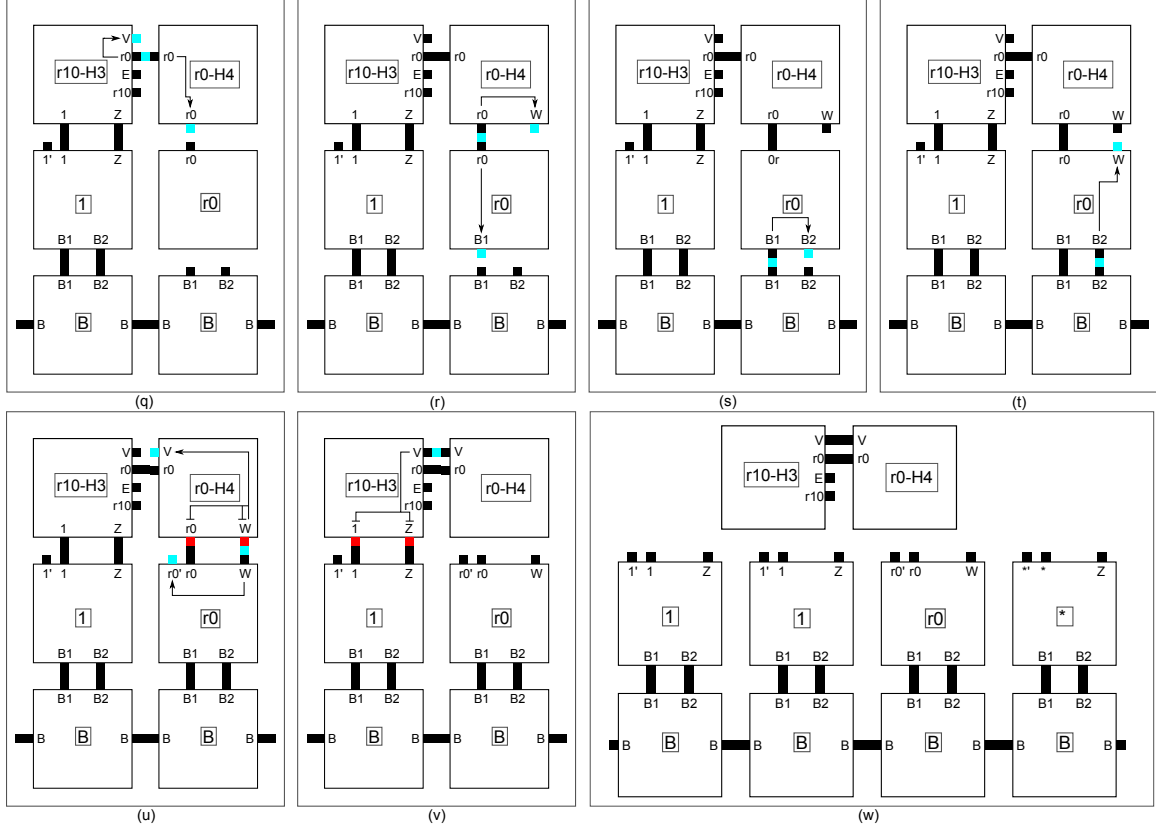


Fig. 22: High-level sketch of the simulation of a transition, part 2 of 2

is ready for next state transition, in a configuration similar to that at the beginning of the transition but with the correct output from the transition which is now complete, and the head in the correct position while representing the new state.

**Growing the tape** In order to simulate a one-way infinite-to-the-right tape with a finite assembly, we simply design the tape so that the rightmost cell always represents a special value, namely \*. Whenever a transition begins which needs to move the tape head to the right, and the destination location of the head currently contains the \* symbol, that situation is “read” by the *rb-H2* tile which forces a \**R* tile to bind to its right, which itself ensures that a new tape cell location with the \* symbol as well as a new backbone tile to hold it in place at the end of the tape both firmly bind. Additionally, the *rb-H2* tile “fakes” the situation of having read a blank (.) symbol so that the location previously occupied by the \* is treated as though it was occupied by a ., which - coupled with the new \* symbol one location to the right - is logically identical to adding a blank tape cell on the right end of the tape. Since this occurs whenever the head tries to move to the end of the tape, it simulates an infinite tape.

**Junk assemblies** During each transition of  $M$ , several junk assemblies are created when they dissociate from the main assembly representing the tape. These can be seen in Figure 21(f), Figure 21(o), and Figure 22(w), and are also shown in Figure 23. In order to guarantee correctness of the construction and thus the underlying simulation of  $M(w)$ , it is necessary that these junk assemblies cannot interfere with *any version* of the computation (keeping in mind that many such simulations would be occurring in parallel). To ensure this, we have carefully designed the junk assemblies so that every glue on their perimeter must be either **latent** or **off**, with the notable exception of the glues *r1C* and *D* on the west



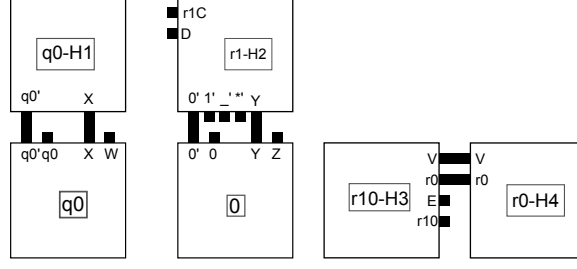


Fig. 23: “Junk” assemblies produced during the example state transition shown in Figures 21 and 22

side of the  $r1-H2$  tile of the middle junk assembly shown in Figure 23. The method of ensuring that specific glues are actually deactivated in a temperature  $\tau = 1$  system is to (1) only send deactivation signals to glues which must be bound at the time the signal is sent and (2) to only send deactivation signals to glues on one side of a bond (i.e. if two tiles are bound together by glue  $x$ , only one of those tiles sends a signal to deactivate its copy of  $x$ ). In this way, despite the fact that signals are processed asynchronously, it must be the case that if a supertile  $\gamma$  is going to detach, all glues on the boundary of  $\gamma$  (whether on  $\gamma$  or the supertile that  $\gamma$  is bound to) which have been sent a deactivation signal have actually already turned that glue **off**. Furthermore, since it is often the case that sets of glues must be turned **on** with the expectation that only one of them will ever bind (e.g. as in the case when the  $rb-H2$  tile activates one glue for each potential tape cell value to its south), in order to follow (1) above, it is necessary to conceal the unused **on** glues from any potential interactions. This need to segregate such glues is the reason for creating junk assemblies of size 2 (and 3) rather than of size 1, and is a technique utilized in all of the constructions which make use of glue deactivation within this paper.

Following the above design techniques makes it necessary that for the leftmost and rightmost junk assemblies to detach, they must be completely inert and unable to bind to any other supertiles in the system. The middle supertile, with the  $r1C$  and  $D$  glues still active, on the other hand, ensures that it cannot bind to any other supertile by utilizing a form of geometric hindrance. Essentially, the  $r1-H2$  tile can only ever dissociate if it does so with a tape cell tile attached to its south. While the  $r1C$  glue on its west would otherwise be able to attach to a  $qa-H1$  tile that is currently bound to the tape using the eastern  $rbC$  (in this case  $r1C$ ) glue, an  $H1$  tile can only ever attach to a tape cell tile after an  $H3$  tile dissociates from its north. The dissociation of the  $H3$  can only happen if it does so while connected to an  $H4$  tile, and the  $H4$  tile could only dissociate along with the  $H3$  tile after “filling in” the new tape cell tile. Thus, for the western facing  $rbC$  glue on an  $H1$  tile to be available for binding, it can never be the case that a junk supertile like that pictured in the middle of Figure 23 would be able to bind: it is too tall (the necessary space for the southern tile would already be occupied by a tape cell tile).

Finally, in the case where the tape is grown one cell to the right there is a fourth type of junk assembly. This junk assembly is simply the middle junk assembly of Figure 23 with a  $*R$  tile bound to the east of the  $rb-H2$  tile by both  $**$  and  $*R$  glues (for a total size of 3 tiles). However, the southern glues of the  $*R$  tile must be deactivated before dissociation of the junk assembly, and therefore no new glues in the **on** state are added. As this is the only other producible type of junk assembly, this means that all junk assemblies are fully inert and unable to bind to any other supertiles, and thus maintain the correctness of the simulation while also never growing larger than size 3.

**Correctness of Theorem 3** It has been shown that the STAM system  $\mathcal{T}_{M(w)}$  which simulates  $M(w)$  correctly simulates the behavior of TM  $M$  on input  $w$ . If  $M(w)$  halts, special  $qa-H1$  tiles for  $q \in \{q_{accept}, q_{reject}\}$  (previously undiscussed) will attach in the head location which halt further assembly (by containing no other **on** or activatable glues), thus creating a terminal assembly of size  $> 3$  (since there must be at least one tape cell tile representing a 0, 1, or  $-$  and the tape cell tile representing  $*$ , along with their backbone tiles). If  $M(w)$  does not halt, the supertile representing the tape will never be terminal - only the junk assemblies will be terminal - and thus there will be no terminal supertiles of size  $> 3$ . Furthermore, it is a temperature  $\tau = 1$  system, all junk assemblies remain at size either 2

or 3, the signal complexity is  $O(1)$  (specifically, 6 due to the maximum number of glues on a tile side being the 6 on the west side of  $rb-H2$ ), the fuel efficiency is  $O(1)$  as every transition produces exactly 3 junk assemblies, each with size  $\leq 3$ , and adds at most 4 new tiles to the tape assembly (the two new tiles for the tape cells which have swapped the head position, plus possibly two more tiles if the tape was grown by one symbol to the right), and tile complexity  $O(|Q|)$  since the tape alphabet and all other glue values are a constant size set and each tile template can be used to generate at most a constant number of tiles for each transition in  $\delta$  and there can be at most  $O(|Q|)$  transitions in  $\delta$ .

## A.6 Section 6: Self-Assembly of the Sierpinski Triangle

### A.7 Section 6.1: Weak Self-Assembly of the Sierpinski Triangle

The full tile set which implements the construction for the proof of Theorem 4, the weak self-assembly of the Sierpinski triangle, is shown in Figure 24.

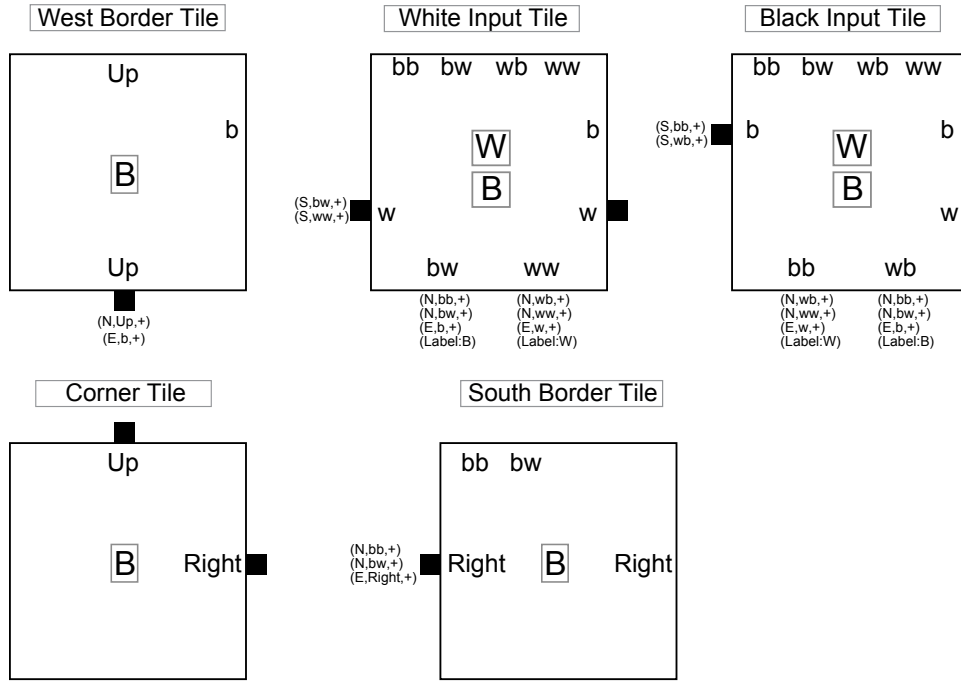


Fig. 24: This set of tiles weakly self-assembles the Sierpinski Triangle.

### A.8 Section 6.2: Strict Self-Assembly of the Sierpinski Triangle

*Proof (Proof sketch).*

The 19 active tile types which compose  $T_{2\Delta}$  are depicted in Figures 25, 26, and 27. The only tiles which can initially bind to each other are those of type  $S$  with  $V1$  or  $S$  with  $H1$ . These initial bindings initiate a series of glue activations which can, using tiles of the types shown in Figure 27, tile each of the blocks corresponding to the points of the positive  $x$  and  $y$  axes in  $S_\Delta$ , namely the points  $\{f(x, 0) \mid x \in \mathbb{N}\} \cup \{f(0, y) \mid y \in \mathbb{N}\}$ . Arbitrarily large portions of the axes can form without the need for any tiles filling in the “interior” blocks (i.e. points  $f(x, y)$  where  $x > 0$  and  $y > 0$ ). However, no interior block  $f(x, y)$  can be fully tiled until blocks  $f(x, 0)$  and  $f(0, y)$  have received tiles. In fact, the

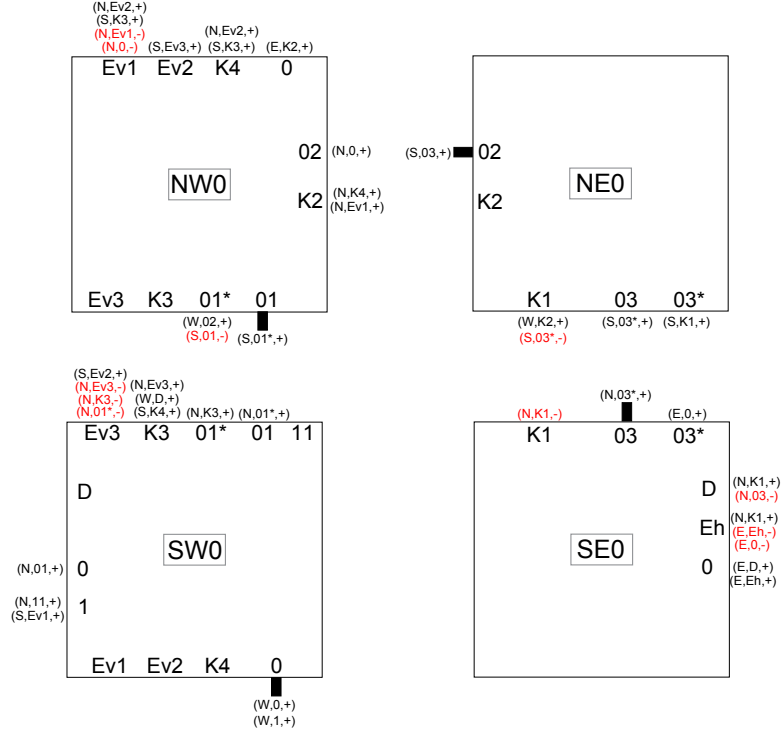


Fig. 25: First group of tile types which strictly self-assemble the Sierpinski triangle

first tile placed in  $f(x, y)$  is always  $f(x, y)_{00}$ , and for interior blocks this tile always attaches first to  $f(x, y - 1)_{01}$  (that is, the north-west tile of the block immediately to its south). For the rest of the discussion, we will only be referring to interior blocks unless explicitly stated.

The first tile to attach in a block is either  $SW0$  or  $SW1$  depending on whether or not the output from the block to the south is a 0 or 1, respectively. (An example of the assembly process can be seen in Figures 28-30.) Once it also binds to the block to its west to receive its second input, it is able to determine the correct output for that block and activate binding glues on its north which specify the identity of that entire block as either 0 (in a white position) or 1 (in a grey position). If it is a 0-block, the rest of the block fills in with the  $NW0$ ,  $NE0$ , and  $SE0$  tiles, in order. ( $NW1$ ,  $NE1$ , and  $SE1$  for a 1-block.) Note that either type of block can have either a  $SW0$  or  $SW1$  tile in its southwest position (the 0 and 1 merely corresponding to the identity of the first input), since the first input is not sufficient to determine the identity of the block.

As blocks fill in, determining their identity as 0 or 1 blocks and then presenting that output to their north and east, allowing further block assembly, those that are 0 blocks also activate glues on their output (north and east) sides which are used to detect that they are bordered by a 1-block and begin the dissociation process. When a 1 block forms in the block  $f(x, y)$ , it must be the case that one of the inputs to the tile in position  $f(x, y)_{00}$  is a 0. If  $f(x, y)_{00}$  is of type  $SW0$ , that input was to the south, and if it is  $SW1$  then that input came from the west. Once both inputs have been determined, such a  $SW0$  ( $SW1$ ) tile will activate a glue which initiates a  $Ev1$  ( $Eh$ ) signal to the south (west). This signal will be received by the bordering 0-block and cause it to dissociate from the 1-block and pass the signal further into the block and the entire white region. The combination of  $Ev$  and  $Eh$  signals which pass through 0-blocks in a white region bounded by 1-blocks (note that they must be completely surrounded by 1-blocks since the order of block growth is strictly up and to the right and those signals are only initiated from the south and east sides of 1-blocks) cause them to separate into vertical chunks which can each dissociate as single supertiles. The design of the signal propagation and glue deactivation is such that the junk supertiles can only dissociate as supertiles which are bounded on both the north

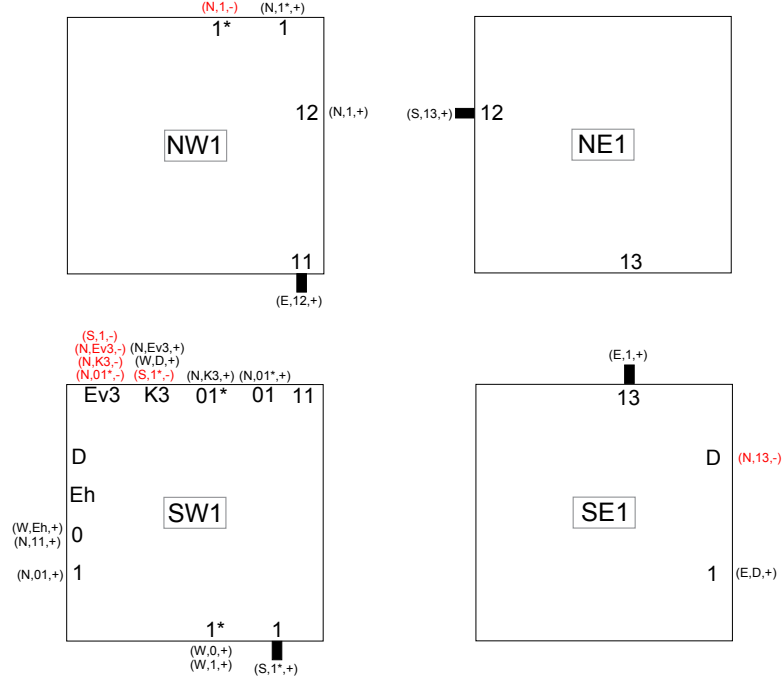


Fig. 26: Second group of tile types which strictly self-assemble the Sierpinski triangle

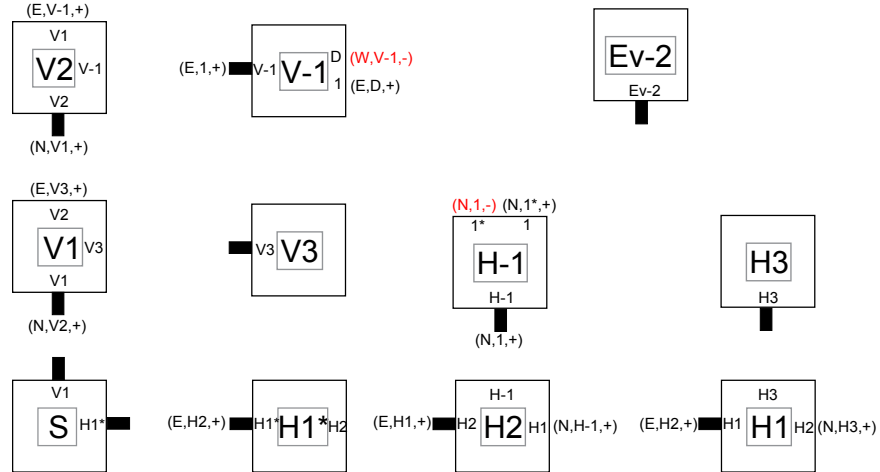
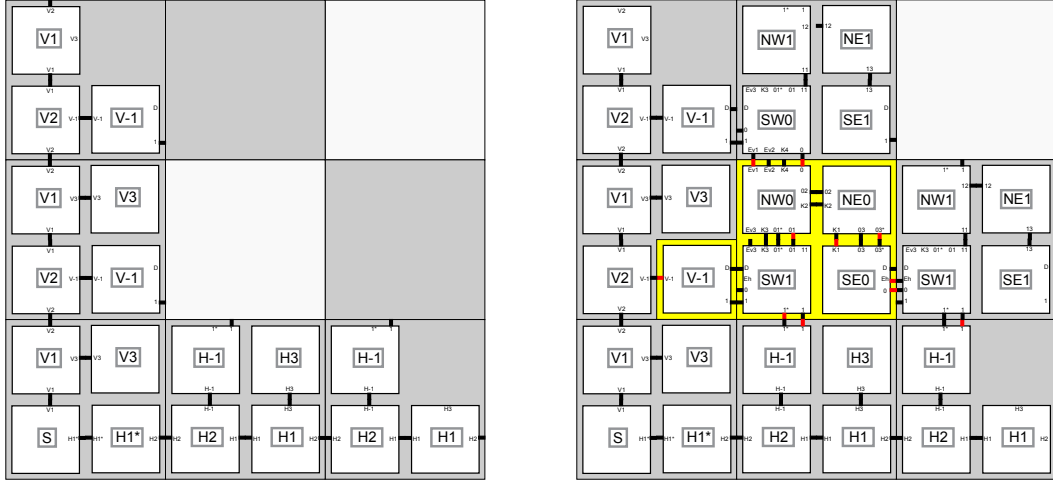


Fig. 27: Third group of tile types which strictly self-assemble the Sierpinski triangle. With the exception of the tile type labeled ‘Ev-2’, these tile types tile the locations along the positive  $x$  and  $y$  axes, which are all included in  $S_{\Delta}$ .

and south by 1-blocks (although they may only be 1 block wide and thus bounded on the east and west by 0-blocks). This fact is extremely important to the process which further breaks apart those junk supertiles into constant sized pieces but also ensures that they will always be broken apart and never interfere with further growth of the assembly representing the Sierpinski triangle.

The first phase of dissociation breaks the white regions into vertical columns which were previously bounded above and below by grey portions. (The process by which some example junk assemblies are



(a) The beginning of the formation of the axes of the Sierpinski triangle (which can continue arbitrarily far without the non-axes locations being tiled). Since the scale factor is 2, the  $2 \times 2$  squares which are included in the pattern are shaded with a darker grey.

(b) The Sierpinski triangle at the point where the first dissociation becomes possible. The tiles in the area with the yellow background will dissociate as one “junk” supertile.

Fig. 28: The initial stages of the strict self-assembly of the Sierpinski triangle.

broken down - including some partial re-growth with the addition of singleton *NW0* and *SE0* tiles - into latent, constant sized junk, is shown in Figure 31.) This means that the bonds between the white and grey regions are broken on the north and south, and also the bonds on the left and right of each column of blocks (both with white and grey blocks) are broken. The careful design of the signals which cause the dissociation of the white regions, always beginning from grey regions to the north and east and initially only causing dissociation with bordering grey blocks to the north and south (and not other white blocks to the north and south of blocks in potentially large white regions), ensures that when junk assemblies are able to dissociate that the glues on their borders which remain active do not allow them to re-attach to any other portion of the growing assembly except for those which have no blocks formed to the north of them. Additionally, if they are larger than 6 tiles they are guaranteed to have an *NW0* tile in the northwest corner of each block in their northernmost row, and that *NW0* tile will have an active *Ev2* glue on its north. Since any position in which these junk assemblies can re-attach cannot have blocks to their north, the *Ev2* glue will allow an *Ev-2* tile to attach, and this attachment will initiate the second phase of dissociation in which a chain of signals cause the junk assembly to break up into fixed height portions. Any re-attachment of the junk assemblies to the growing structure will therefore be temporary and also not able to cause incorrect growth by initiating invalid signals. Thus the junk assemblies are broken into fixed width and height portions in two phases.

It is notable that junk assemblies can allow singleton *NW0* and *SE0* tiles to attach at some point during their break up into constant sized pieces, but that only these single attachments are possible and no additional signal activation is possible which would allow the junk assemblies to make further attachments. Another interesting aspect of the dissociation process is that white blocks which have grey blocks bordering them on their west side will cause a single tile of those grey blocks to dissociate with them. The newly formed hole in the grey block will be re-filled by either an *SE1* or a *V-1* tile. The reason for this is to “hide” the active 0 glue between the original *SE1* tile and the *SW1* tile to its east, which could allow incorrect binding of the junk assembly. In fact, the need to hide active bonds which cannot be guaranteed to be deactivated in this asynchronous model is the reason for the complexity of the dissociation process and the size of the final, terminal junk assemblies.

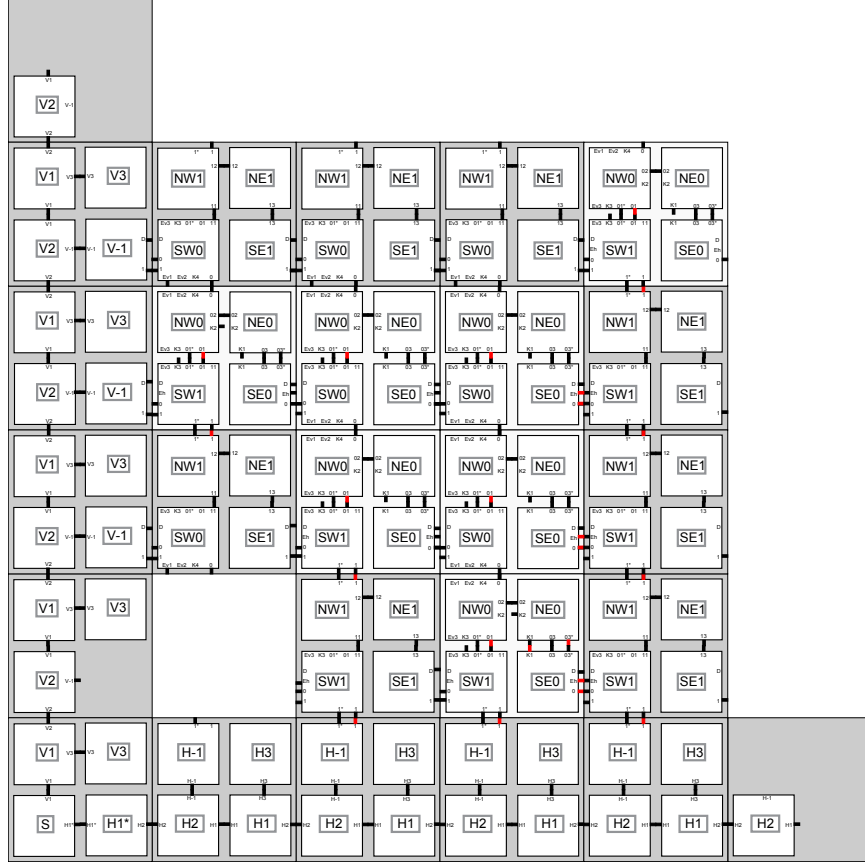


Fig. 29: After weak self-assembly of the first three stages of the Sierpinski triangle plus an extra row and column to provide the necessary signals which initiate the dissociation of the interior white portions. Those signals have only partially propagated into the white portions in this figure.

Through this process in which weak self-assembly of the Sierpinski triangle proceeds until white regions are surrounded by grey blocks, and then those white regions are forced to dissociate as arbitrarily large junk assemblies which are then further broken down into constant sized junk assemblies, all the while being guaranteed not to cause incorrect assembly by binding to the still growing structure, provides the correct strict self-assembly of the Sierpinski triangle at scale 2.



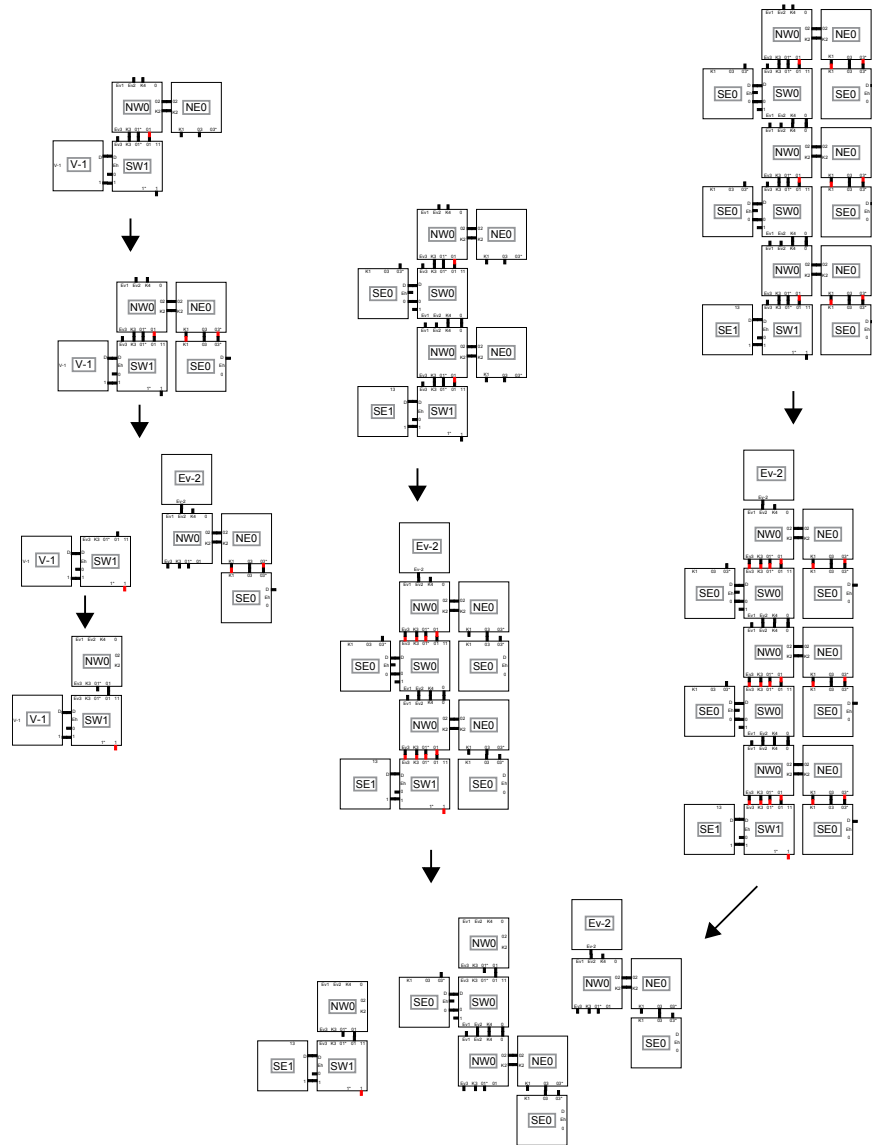


Fig. 31: The “junk” ejected from the first three stages and the pathways along which it is broken down following the attachment of  $Ev - 2$  tiles. Note that at some points, singleton tiles can attach to the junk assemblies. However, all junk assemblies are guaranteed to break down into sizes of 3, 4, or 6 and become terminal.